TISHUG NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 8, Number 12 December, 1989

Registered by Australia Post - Publication No. NBH5933

P.O. Box 214, Redfern, New South Wales, Australia, 2016 $3

TIsHUG (Australia) Ltd.

TIsHUG News Digest

December 1989

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

The Board

Co-ordinator
Dick Warburton (02) 918 8132
Secretary
Terry Phillips (02) 797 6313
Treasurer
Rolf Schreiber (042) 84 2980
Directors
Robert Peverill (02) 602 4168
Russell Welham (043) 92 4000

Sub-committees

News Digest Editor
Geoff Trott (042) 29 6629
BBS Sysop
Ross Mudie (02) 456 2122
BBS telephone number (02) 319 1009
Merchandising
Steven Carr (02) 608 3564
Publications Library
Warren Welham (043) 92 4000
Software library
Terry Phillips (02) 797 6313
Technical co-ordinator
Lou Amadio (042) 28 4906

Regional Group Contacts

Carlingford
Chris Buttner (02) 871 7753
Central Coast
Russell Welham (043) 92 4000
Coffs Harbour
Kevin Cox (066) 53 2649
Glebe
Mike Slattery (02) 692 0559
Illawarra
Geoff Trott (042) 29 6629
Liverpool
Larry Saunders (02) 644 7377
Northern Suburbs
Dennis Norman (02) 452 3920
Sutherland
Peter Young (02) 528 8775

Membership and Subscriptions

Annual Family Dues	$25.00
Overseas Airmail Dues	AUS$50.00

TIsHUG Sydney Meeting

The next meeting will start at 12 pm on 2nd of December at Woodstock Community Centre, Church Street, Burwood.

Printed by
The University of Wollongong
Printery

Index

They're off

by Geoff Trott

We had a good day at the full day tutorial, even though the venue was changed at the last minute. I was busy with dead consoles and Lou seemed to be constantly conversing with members who were interested in the many projects he has originated this year. The other groups had lots of room to attract customers and there was very little interference with each other. It seemed to be a very good place to meet. I had a few interesting problems with consoles. The first came from the console repair group where they had diagnosed video memory problems and changed two chips without any improvement. Working out the patterns shown by the console tester showed that at least one chip was bad and possibly more. Removing that chip and then running the console tester actually seemed to give a slight improvement and replacing the chip fixed the video problems. That video RAM chip must have been interfering with the operation of other RAM chips. I then tried to run my diagnostic module and it did not appear on the menu. Clean the edge connector, no help but try it in another console and all is OK. Ran the console tester again and the non-existant GROMs were producing non-zero checksums. Look at the circuit which is supposed to pull down the data bus to the GROMs when reading them and it is not working. Trying to locate a resistor to put the oscilloscope lead on to check one of the signals and the resistor is not there! In its place is a diode, never been touched, installed in the factory. Replaced it with a suitable resistor and all was well. The problem would only show up on some modules as Extended BASIC was OK. That was a strange one!

continued on page 30

Co-ordinator's Report

by Dick Warburton

What motivates a person to join a computer club like TIsHUG, and stay in it? How did you join? Did someone bring you along? Did you come out of curiosity? Where did you find out about the club? Did you need help? Did you get help? Did you find the members friendly? I suppose there are almost as many reasons as there are members. I know that I came along because I felt I was so ignorant about computers, and I wanted to learn to use one. I found the basic manuals did not meet my needs, and I came looking for help. I suspect that many of us join a group to be helped to achieve our computer goals. At the time I joined, TI was ceasing to market the TI99/4A, and I was concerned about support and future problems. I bought a TI99/4A by accident. I had decided to learn a bit about computers by learning to use one. I asked my son (the only computer literate member of the family), what to buy. He went away, read the magazines, and told me there was only one real choice, if I wanted a real computer. Fortunately the price dropped sufficiently for me to afford an expanded system. Unfortunately, my wife likes to write books, so we bought TI-writer, and I had considerable difficulty actually using the TI99/4A at any time. However, because she was so busy she bought another system, and I took over the TI99/4A. Well, that marked the beginning of a different relationship in the house. Does your wife or husband use your TI99/4A? Whatever happened, it triggered a more active membership in the club. I have gained far more from belonging to this group, than I could ever hope to give back. I found a new world opening up to me. I found an exciting new tool which could do all manner of things if one put the effort in to program it. I laboriously wrote some simple programs which actually did what I wanted. For example punting programs, but alas, computers are no better than the program we devise. I suspect if I had relied on my programs to make a quid", that I would be broke by now. What gradually dawned on me, was the tremendous satisfaction, of learning to do a variety of computer things for myself. For example, to open the console and change a chip, to diagnose simple faults. I still find it difficult to realize that I can now do these sorts of things. I have gained some basic knowledge about how computers work, how to use tools, how to handle electricity. But most of all, I have met a great group of people, some of whom have become friends. I suppose now that the needs I had when I joined, have changed. I can now do many more things for myself, however I want o stay in this beaut group, because of the friendship and mutual help we give each other. I suspect that there are still people outside the club who need to join a group for the sorts of reasons we had. They need our help. They need to know that the club is available first of all. They need to have some of their needs met when they come. They need to be made to feel welcome, to become part of this ongoing group. We will be trying to reach old members, and recruit new ones over the next year. If you enjoy your membership, if you have gained something for yourself from belonging, then help the club next year by giving some of your time and skill back to the club. We need volunteers to seek out new members, to publicize the club, to help our newer members, to repair consoles, to accept office in the club.

1990 looks like being a good year for TI99/4A users, particularly from the hardware point of view. Firstly, because TI99/4As are out of fashion, they are now really cheap. If you have an unexpanded system, now is the time to think seriously about bringing it up to a really useful standard. Disk drives are plentiful. Second hand drives will cost from 15 dollars to 120 dollars. You can buy or make a power supply fairly cheaply. People are selling peripheral expansion boxes quite cheaply, when you consider how strong and reliable they are. We are planning all manner of projects for 1990. We are purchasing 8K and 32K static memory chips. We will have some printed circuit boards made up, for example, for RAMdisks if there is sufficient interest. We are hoping to develop an EPROM version of the RAMdisk, with all our important software in EPROM. We will have a simple expansion card for those who cannot get a PE box. There are new cards coming from overseas. In December I want to start a project group to actively develop and complete new TI99/4A projects. There is sufficient interest already to start. It is hoped to have kits of parts available to members to allow them to complete these projects. We have not yet decided where to meet. If the interest is sufficient, we will break into two or more regional groups. I have found that my greatest enjoyment with the TI99/4A came when I completed my first RAMdisk successfully. Do not miss out. Have a go at one project at least in 1990. I have at least four projects I want to complete for my own satisfaction. come and join in the fun. Hopefully we will liaise with other groups, and work, co-operatively. Make sure you get some real satisfaction from your membership in 1990. Join in the fun and the friendship, and help others to enjoy it too.

See you soon

Dick Warburton. o

Letter to the Editor

Dear TIsHUG,

I have finally decided to shake the dust off my system and do something with it. Living in the country, far away from other TI99/4A users, I have not put my TI99/4A to much use recently. I thought it would be better off in the hands of someone who would use it. My full system includes:

Silver and black console, power supply and VHF (was converted from UHF) modulator
32K memory expansion (TIsHUG kit)
Peter Schubert RS232/modem
MiniMemory plus MiniWriter
Touch Typing Tutor
Terminal Emulator II
Extended BASIC
Personal Record Keeper
Speech Synthesizer

All these items are fully functional and have not received much use. I also have some games on tape and a number of books of TI99/4A programs etc. If TIsHUG is interested in this system I will sell it for $200 and I will pay postage and packing from here. I thought it may provide backups and spares which could keep the dedicated user going longer. If the group is not interested, would you please run an advertisement in the next club magazine. I am only interested in selling the whole system, not parts of it and to anyone apart from TIsHUG I would ask $250.

I feel the main reason I have not put the system to use is because I have not got a disk drive. At the end of 1987 I intended to purchase an expanded system second hand, but found I could buy a new computer, disk drive and software for about the same price.

It is with some regret that I have decided to sell, as I have used the TI99/4A system:

1. to learn the rudiments of word processing;
2. as a teacher's mark book using Personal Record Keeping;
3. to learn the "basics" of BASIC programming;
4. to entertain my children, particularly with the speech synthesizer; and
5. to use the BBS and the Department of Education's keylink service.

I now realise that availability of software and compatibility are more important than the quality of the computer, unless you are a dedicated enthusiast.

I also appreciate TIsHUG. Without the monthly newsletter, which I invariably read, I would have stopped using my TI99/4A three or four years ago.

continued on page 13

Secretary's Notebook

by Terry Phillips

The December meeting will mark the end of another year's activities, and with a fine day it promises to be a meeting where members and their families can get together and relax with party food and drink provided by the group. So make sure you come along and have a chat to your fellow members.

Inside this issue you will find a nomination form which can be used to make nominations for Directors' positions. As you are aware the Annual General Meeting is held on the first Saturday of February each year, with the 1990 date being the 3rd, so mark it in your diary now. We are hopeful of again having the use of the Burwood RSL Club auditorium to conduct our 1990 AGM. If nominating a member for a Director's position, make sure you get a seconder and that the member nominated consents to his nomination. Nominations close with the Secretary at 8pm on the 13th January, 1990. If the required number are not received by that date then further nominations can be made at the AGM.

Feedback I received on the day would indicate that most who attended the November full day meeting enjoyed the change of venue. Certainly it gave us much more opportunity to spread out and conduct the various activities in separate rooms. Unfortunately there was little time to publicize the venue change. Our booking for Shirley House fell through and Woodstock main building was unavailable for more than 2 upstairs rooms. If you did not get your TND before the meeting, and were unaware of the venue change then sincere apologies.

At the meeting I was kept busy all day doing disk copies and the time just seemed to fly by. I was also booked to give a tutorial on TI-Base for both morning and afternoon but during the morning I only had one attendee. I did not do the afternoon session as the committee meeting was in progress. Shane Ferrett took over for me. Thanks Shane.

I hope everyone got onto that great bargain in Xidex disks at the meeting. Great work on Dick Warburton's part in sourcing out that supply. I do not think you could buy cheaper disks anywhere.

Dick will probably cover this also in his column, but the news is we have had to give up on our bulk importing of MICROpendium. Chief reasons being are declining sales together with the large stock of back issues we have which are tying up club funds. If you know of anyone who would like to buy some then see Stephen at the meetings as I am sure some deal can be arranged.

There are available 4 sets of Asgard News, Volume 1 Number 4 and Volume 2 Number 1. If you would like to purchase these then see me at the next meeting. Subscription is $17 for 4 issues and the next issue should be received soon.

We only have two new members to welcome this month. They are:

Sylvia Kwok of Chatswood, and
Garry Hughes of Kurnell.

A big welcome to you both and I hope you can make it to some of the main or regional group meetings.

Copies of TNDs being mailed to Tony Imbruglia and David Rivett are being returned marked "Left Address". If anyone knows there whereabouts could they let me know please.

That is it for this month, but please give the important matter of nominations for Director some thought over the next few weeks. o

TIsHUG Software Column

by Terry Phillips

Well I do not know how many disks were copied at the November meeting, but it sure felt like I was busy all day, and I hope that everyone got what they wanted. Thanks go to Ross Mudie for the loan of the disk copier. It sure speeds up the job.

On the software scene there are two major items to talk about this month.

The first being the release of TI-Artist Plus (or TIAP for short). What we have here is a complete revamp of the old TI-Artist. The new version comes on 3 disks and supports Pictures, Instances, Slides, Fonts, Vectors and Movies. A 38 page manual accompanies the package and gives details on all these capabilities. A couple of the users of the old TI-Artist who tried out this version at the last meeting were very impressed with what it can do. If you are a current owner of the older version then you can get the upgrade for about $15 and copies should be available at the December meeting. Price is subject to import costs. If you are not already an owner then TIAP can be purchased for approximately $25. Dennis and Chris Faherty of Inscebot Software are great supporters of the TI99/4A community, so please support them, and your club, by buying their products. Some of the better TI99/4A software is currently being marketed by this talented father and son team.

The second item is a new revamped version of TI-Writer, titled TI-Writer Version 4.0. This is Fairware with a suggested contribution of $10 to RAG Software (R A Green), 1032 Chantenay Drive, Gloucester, Ontario Canada. The software supports a host of Editor and Formatter improvements together with new format commands. A simple QQ will exit from the Editor without any further key presses. I like it and will have some copies available through the shop at the December meeting.

If you are looking at some disks from the library for Christmas, then give me a list together with a few dollars for postage and I will make every effort to get them back to you by that time. o

continued from page 30

PUG Peripheral, July '89. Zeno board (console internal peripheral expansion) has been completed, TI-Artist for beginners Vol3, table of TI99/4A key character codes, high resolution graphics for the 99/4A, review on kindermath and TI-Sort, reminder that Star NX1000 printers with ROMs higher than V1.31 will not work with the TI99/4A and how to fix "Macflix" in order to get a 1:1 aspect ratio on printout. PUG Peripheral, Oct '89. "Interface Standard Design Guide for the TI99/4A Peripherals", Zeno modified console with internal clock, speech, 32K, Extended BASIC, Editor Assembler, DM2, TI-Writer together with pause, reset and GROM disable switches! Another new release is a "RAMBO" (random access memory bank operator) PCB which allows Horizon RAMdisk to be partitioned as CPU memory up to the capacity of the RAMdisk, more Geneve operating information, High Resolution Graphics continued, console hardware debugging hints covering lockup, module errors and Joystick port errors, Printers #2 by J. Willforth on using control codes from BASIC, fix for Funlweb V4.0 which allows marking files with FCTN[7] in the Formatter, changing defaults in the Formatter, how to deactivate "out of paper" feature to allow printing close to bottom of single sheets of paper or for addressing envelopes with a printer and lastly sorting numbers in Forth.

TIdbits Oct '89. Has information about the Chicago faire, Bill Gaskill's Four-A/talk, a review of TI-Sort and 9640 news by Berry Miller. o

New Programs and Games

typed by Larry Saunders from Asgard News

Tris

An extremely addictive mind teaser! In Tris you must rotate and move colourful , falling shapes to fill in the holes in the bottom of the screen. Complete rows disappear but incomplete ones just cause the screen to fill up! Simple to play but difficult to master, Tris will challenge and amaze for hours. This novel game is based on the popular Russian program that perpetually tops the best seller list for IBM and Apple software, but it has better sound effects and colour then any version ever produced! (this is not the version that is on this BBS).

Karate Challenge

Evil Dragons have taken over the minds of the students and teachers at your Karate Dojo. In this fast-paced action game you must fight your former friends and teachers in order to get a chance to defeat the dragons in mortal combat. The amazing graphics of this game depict the many kicks and punches you can throw at your opponents in order to save the day.

Mission Destruct

Evil robots have taken over the moon! You are on a mission to destroy as many moon base reactors as you can before the waves of Death Droids, Space Mines, and the evil Draks overtake your frail craft. This is a truly fast, arcade style game that has great sound effects and fantastic graphics.

Legends II

You are ship-wrecked with fellow adventurers on the island of Femble, leagues from home. You have to explore this vast island, fight unknown monsters, explore a city and plumb the depths of dungeons in order to raise enough dough to buy passage home.

Zoom Flume

You are in your bathing suit at a water park. You only have so much money to spend, so spend it wisely!

Witch's Brew

A witch has let loose spells across the land. Can you find the witch and remove the spells without becoming part of one?

Wizard's End

A multi-player game with monsters, spells and treasures in the traditional Dungeons and Dragon's mold. (For the advanced players only.)

Oliver's Twist

Can you return the 15 treasures to their rightful places without disturbing the ghost of King Oliver? He will kill you if he disturbs you or if you take too long.

Disk of Pyrates

A fascinating 4 disk package. Disk of Pyrates is a veritable cornucopia of Pyrate artwork, Pyrate games, Pyrate music, Pyrate animation and Pyrate history.

Music Pro

This is a Music Processor program that allows you to type on a staff directly from sheet music, then compile and play the music.

The Animator

A complete frame by frame animation package. It is an extensive program that allows you to create multiple animation sequences one frame at a time, and then script them together in a complicated animation sequence.

Pix Pro

The "super converter" to meet all your graphics needs. It will convert almost any format to almost any format.

TI-Writer Version 4.4

This program has had the works done to it. It is faster, a lot of new commands, for example .PC (printer control) or CTRL',' (will go to start) or CTRL'.' (will go to end of the document. It also has an install program that customizes your editor and formatter.

TI-Writer will be at the next TIsHUG meeting at the word processing clinic along with almost every other word processing program for the TI99/4A. o

TEXPAC BBS Version 4F

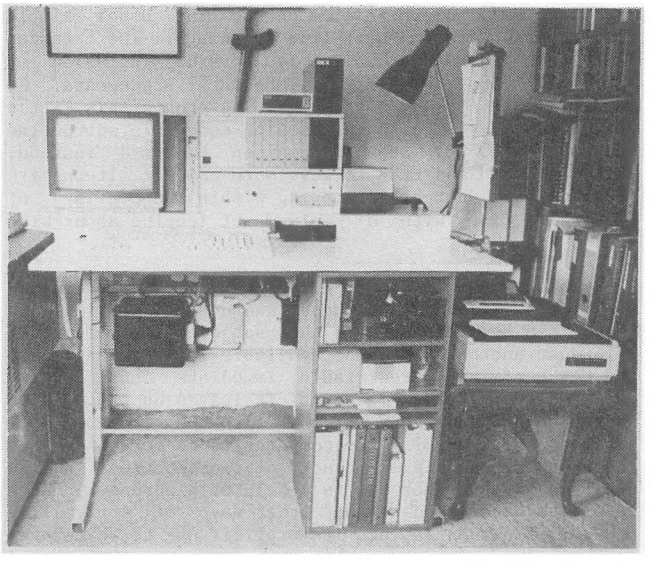
by Ross Mudie

User BEN was having trouble loading a file through the mail system. On examination I found that it contained strings of ASCII 12,13, that is a form feed and carriage return. The BBS was set up to discard any character outside of the range of ASCII 16 to ASCII 127, so the ASCII 12 was discarded and the ASCII 13 was the only byte in the string which dropped the sender out of mail (a carriage return on its own).

Version 4F allows characters from ASCII 0 to ASCII 15 to pass on to the disk for storage as mail with the exception of ASCII 8 which is backspace, ASCII 10 which is line feed and ASCII 13 (carriage return) which signifies end of a line if other characters exist in the line or end of mail on its own.

Version 4F also changes the operation of backspace. Previously the backspace character just acted on the string in the BBS editor, doing a destructive backspace along the string, leaving the string intact on the user's screen. The new version wipes out characters on the users screen by echoing ASCII 8,32,8 each time an ASCII 8 is received. This moves the cursor back one space, then writes a space which also moves the cursor one to the right then moves the cursor one character to the left again.

If anyone has any problem with the expanded characters available please let me know. o



Ben Takach's highly modified TI99/4A system

Extended BASIC Tutorial

by Tony McGovern, Funnelweb Farm

V. Extended BASIC STYLE WITH SUB-PROGRAMS

Let us now stand back a bit and look at the best way to construct Extended BASIC edifices. Assume at this stage that we are in the process of developing a program, but not yet to the point where scrunching program length has become important. The first thing to note is that by giving the sub-programs good descriptive names you have already gone a long way to making your program self-explanatory.

How big should individual sub-programs be allowed to get? After all, one of the reasons for using them is to break up big programs into manageable hunks. We will use the term 'line' to refer to a multi-statement Extended BASIC line identified by a line number. My own prejudice is that, except in special circumstances, sub-programs should be no more than about 10 lines long, and mostly rather less than that. What makes an exceptional circumstance? An obvious one is in title blocks, like that in SIMPLIST which was left as an almost bare stub. A full version would provide graphics and advice screens, which can be tediously long to write, but contain very little in the way of branching decisions or variable assignments. Another example is where a familiar routine, that already works, is used with little variation as in COLIST where the disk directory routine from the Disk Manual is incorporated as a subprogram with only minor changes. In any such situation where long sub-programs are justified, the lists of parameters passed will be short or non-existent.

The other extreme is short one or two liners which are frequently CALLed for small special tasks, more or less your own customized extension of the built in set of sub-programs. In the middle there are middle length sub-programs with extensive parameter lists and the logic which carries the burden of program flow.

Some sub-programs may be CALLed only once from within another sub-program but are of value in making your code easier to read and modify. These are associated with the branching of program flow by means of IF..THEN..ELSE statements. In either TI BASIC or Extended BASIC , FOR-NEXT loops may extend indefinitely with NEXT acting as delimiter. Unfortunately in extending BASIC to Extended BASIC, TI did not provide an "ENDIF" statement as in TI-Forth, but only the 'endif' implied by the end of a Extended BASIC line. This means that any alternative actions determined by the IF.. condition have to fit within that Extended BASIC line or involve a GOTO somewhere else unless the usual simple drop through to the next line is enough. The Extended BASIC manual already explicitly forbids inclusion of FOR..NEXT loops within IF..THEN..ELSE statements. No doubt you are already used to getting around this little deficiency by placing the looping code in a subroutine and using a GOSUB. Sub-programs can be used instead, following THEN and ELSE to give more complex alternative possibilities, but still staying within the confines of a single line with a minimum of leaping about with GOTOs.

This brings us to the subject of the 'dreaded GOTO'. A great deal of heat, and not necessarily much light, has been expended on this subject. It is after all just another statement available in many languages, and has perfectly predictable immediate consequences. The real objection is that it leaves no trace as to where the program "came from". At the machine code level, jumps enable the computer to do more than just chomp along a single track of instructions. The question is whether it is a help or hindrance in high level languages, and whether other ways of controlling program flow can replace its explicit use to advantage. TI-Forth does without it, but that most procedural of languages, TI-LOGO, still finds it useful. Pascal tries to do without it. What we do have is Extended BASIC, and Extended BASIC cannot do without GOTOs. If anything should be considered as reprehensible in a high level language, it is any need to provide PEEK and POKE.

The great weakness of GOTO as a language element is that it is so readily abused, because undisciplined use makes the program code inefficient and hard for people to follow. The genuine message from 'structured programming' ideas is not that BASIC is bad for having GOTOs, but that most BASICs (TI console BASIC is typical) make it necessary for the programmer to exercise real restraint if terrible tangles of GOTOs are to be avoided.

Once you use Extended BASIC sub-programs to chop up a program into small hunks, then you have automatically eliminated great leaps around with GOTOs. All you need then is to remember the comments on using sub-program CALLs as statements in IF..THEN..ELSE and take a little care in laying out the logic flow, you will find it very much easier to debug or develop programs. Backwards GOTOs over more than one or two lines of code, or any forward GOTOs at all, should only occur under the most regular of logical layouts, as in SUB BASICLINE in the SIMPLIST example. Single recursive lines such as in line 620 of SIMPLIST are very effective. It is a pity that the designers of Extended BASIC did not add the "MYSELF" function as in TI-Forth to enhance such constructions.

One last little matter before we go on to other topics. Many languages with local procedures also allow specification of global variables, accessible from any part of the program. Extended BASIC does not allow for separate global variables, and it can be quite tiresome when a parameter defined at the end of one sub-program chain is only needed at the end of another chain, and has to be passed all the way up and down in parameter lists. A way around this is to use the static value feature of Extended BASIC sub-programs .

```
3000 SUB PAGELENGTH(A,B):: IF A THEN C=B ELSE B=C
3010 SUBEND
```

If the write flag is set as CALL PAGELENGTH(1,66) the value 66 is stored in the sub-program local variable C, while CALL PAGELENGTH(0,PL) will retrieve that value into PL. This is clumsier than having global variables, but is also more protected from unwanted interference. Extended BASIC does not enforce any hierarchy of sub-program levels, so PAGELENGTH can be written to, or read from, at any level in the program. The example is for one parameter only, but is easily extended.

Using sub-programs does carry some overhead expenses, both in the size of the program and in the time taken for Extended BASIC to do a CALL as distinct from a simpler GOSUB. Unless you are absolutely desperate for bytes, the benefits of sub-programs outweigh that price and they should always be used liberally in the early stages of program development. The speed argument is a mixed one, as in a long program the extra time cost of a sub-program CALL can be more than outweighed by the savings in time for the interpreter because it has only a short local list of variable names to search instead of a much larger global list.

VI. PRE-SCAN SWITCH COMMANDS

The little supplementary booklet that comes with the current Version 110 of Extended Basic introduces a new pair of reserved words, !@P+ and !@P-. These have the form of a tail remark (Extended BASIC manual p38) and so are ignored entirely by the earlier V.100 of Extended BASIC. If the Extended BASIC interpreter finds an exclamation mark ! outside any DATA string or string enclosed by quotes, it treats the rest of that line as though it were a REM statement. The V.110 interpreter has the added ability to recognize this pair of words beginning with ! as being distinct from normal tail remarks when used as a single word statement. Their use is allowed only at the end of a line so that V.100 just

ignores them, not creating any incompatibility problems between versions, something that TI was always conscientious about. TI then could not let these commands actually do anything! So why are they there ?

The Extended BASIC manual addendum, p7, tells the story. These switch commands allow you to control the operation of the pre-scan through the program by the interpreter: that agonizing time interval after RUN is entered before the program starts executing. The interpreter is grinding its way through your program, byte by byte, ignoring only the messages in DATA, REMs and tail remarks. Other than these there is nothing that it can afford to ignore until it has actually looked at it. The pre-scan sets up the storage areas and lookup procedures for variables, arrays, data, sub-programs and DEFs used by the interpreter as the program runs. Of course once it has set aside space for a variable and its lookup linkages, then it does not need to do it again or even to have to decide it has already fixed it up earlier. The pre-scan switch commands allow the programmer, from a superior vantage point, to turn the pre-scan off and on throughout the program so that it only looks at what it really needs to look at to do its job.

What does the programmer gain by going to all this extra trouble? The most obvious result is a reduction of pre-scan time. This can be significant in long programs. The 6 to 7 seconds for TI Extended BASIC, a 12K program, may still seem long but beats 4 times that. In a later chapter we will see how it can be used to fine tune run time behaviour as well. What price does the programmer pay for these benefits? The necessary penalty is the memory space taken by the extra statements. The hidden penalties, incurred while writing programs, are the inscrutable bugs that may be introduced into the code and the loss of some program checking during pre-scan such as FOR-NEXT nesting.

Let us work our way through the Extended BASIC manual's prescriptions. Some of these help give insight into the way Extended BASIC conducts its affairs. My experience is that some of the restrictions need not be followed strictly as laid down, as long as the essential spirit is observed, while some are absolute, and others are in between. These last are the ones where it is possible to imagine another version of Extended BASIC doing things differently while still being according to the book. This is always the danger in using unspecified properties or "undocumented features". It is not such a problem with Extended BASIC since TI pulled the plug on the TI99/4A and made Extended BASIC a language as dead as Latin. In retrospect this last statement is no longer quite true, and though people continue to use the original Extended BASIC there have been enhanced alternatives, still GPL based using various GROM simulation schemes. There have also been utility programs to do automatically some of tasks we are discussing here and more. What has been sorely lacking has been any published exegesis of just how Extended BASIC goes about its business internally. I have never seen any source code for Extended BASIC, either original TI or reconstructed. I do not know whether TI source has leaked out, or even if it still exists, but if it has it is being closely held, and the writers of Extended BASIC processing utilities do not seem to have felt any urge to share around the details of what they found.

(1) DATA statements :-

The pre-scan locates the first DATA statement and sets Extended BASIC's data pointer for the first READ operation to use. If the first DATA is skipped in the pre-scan, then RESTORE must be invoked before the first READ to set the data pointer correctly. If this is done, the Extended BASIC manual's advice can be ignored.

(2) Variables :-

Each variable must be scanned once, otherwise Extended BASIC will not have it in its linked list of pointers to names and storage locations. This can be the source of some truly evil program bugs, where a syntax error message results from a line of code which looks perfectly correct. The reason can be that injudicious positioning of pre-scan switch commands has left the interpreter with something that should be a variable, but cannot be located as such. Being a non-variable is a much worse fate than merely being set to zero.

OPTION BASE 1 affects how storage is allocated and normally precedes any array references. If hidden from the pre-scan by !@P- then the default 0 will apply.

The manual says that the first occurrence of any variable or array must be included in the pre-scan. This would seem to be necessary for arrays, in the DIM statement, unless you are using the default (no DIM) dimensioning. Simple variables can be pre-scanned anywhere as long as it is at least once. Try the little sample program

```
100 CALL CLEAR :: !@P-
200 I=1 :: PRINT I
300 !@P+
400 I=2
```

Run this program and there will be no problems. Delete line 400 and see what happens. Now you will have a syntax error in a line that by itself is perfectly correct.

(3) Sub-programs :-

The Extended BASIC manual recommends that the first CALL to any sub-program be included in the pre-scan. It would appear that if the first CALL to a user defined sub-program occurs after its own SUB (from within a later sub-program) then the necessary inclusion of the SUB and SUBEND markers suffices.

Built-in sub-programs of course do not have associated SUB statements, so a CALL must be included in the pre-scan if the program is to run normally. Try this example.

```
100 FOR I=1 TO 1000 :: !@P-
200 CALL SCREEN(12)
300 !@P+
400 NEXT I
500 SUB ANYTHING :: CALL SCREEN(3):: SUBEND
```

This will run even though SCREEN is pre-scanned only in a sub-program. Delete line #500 and it will crash if you are running Extended BASIC with the 32K memory expansion. In VDP RAM (console only) it still executes but only at about 1/3 the speed.

What happens if an array is referenced in the parameter list of a sub-program, but not dimensioned until a later sub-program? If you recall the discussion on passing arrays by reference, you will not be surprised to find that Extended BASIC is smart enough to hold over assigning space for the array until it comes across a genuine program reference. Try this little example

```
100 CALL SECOND
200 SUB FIRST(A()):: PRINT A(20):: SUBEND
300 SUB SECOND :: !@P-
400 DIM A(20):: CALL FIRST(A())
500 !@P+
600 SUBEND
```

This program crashes with a syntax error in 400 in SECOND. Now delete the pre-scan commands and the program will run. If you further delete DIM A(20):: in line 400 the program will crash in 200 with a subscript error.

(4) DEF, SUB and SUBEND :-

Do as the book says. Extended BASIC needs these in the pre-scan to set things up correctly. continued on page 13

Techo Time

with Lou Amadio

The last TIsHUG meeting was held at Ryde Infant School. This was a tutorial day with a lot of interesting talks arranged by Craig Sheehan. Once again we were kept busy all day on hardware problems. Geoff in particular had a very late lunch with some tricky console repairs.

Multifunction Cards

There was a lot of interest in building hardware items, in particular the Peter Shubert Multi Function Card. These cards are now available through the club shop for $30 and can be configured with any or all of the following options:

Double density disk controller
RS232 #1 and #2
PIO printer port
32K memory expansion.

We will be supplying most of the chips for the above options and preparing instructions on how to add them. Most of the building will be fairly straight forward except for the DDDC option which will need a CRO to set two pots. This will be done at one of the meetings or by prior arrangement.

Direct I/O Interface PCBs

Double sided printed circuit boards for the interface are now available through Geoff Trott (042-296629) for $35. We are also seeking supply of a plated through version to simplify construction. You may write to Geoff at:

20 Robsons Road, Keiraville, NSW 2500

32K Memory Expansion

by Lou Amadio

The following hardware article is part 1 of a two part series on memory expansion for the TI99/4A console. The purpose of the article is to assist members in choosing a memory expansion options which best fits their long term plans.

STOP PRESS

The club has purchased a number of 8K and 32K SRAMs at very good prices. See the club shop for availability.

The 32K memory expansion is perhaps the most useful hardware addition for the TI99/4A computer. Indeed, if you want to do anything serious with the console, including quality games, then 32K memory expansion is mandatory. The reason for this is that all machine language programs need the expansion memory to run. Not only are you able to run more sophisticated software, but there is a much larger storage area for user data, such as text during word processing, etc.

The TI99/4A console has 16K of memory in its standard configuration. This memory is controlled by the Video Display Processor (TMS9929A) and is used to run BASIC and Extended BASIC programs as well as some housekeeping functions. The 16K of VDP memory cannot be expanded any further.

The TI99/4A, however has a second processor, (TMS9900) which is called the main CPU chip. In a standard console the CPU controls only 256 bytes of RAM (user memory). This RAM can be expanded by an additional 32K for a total of 48K RAM. Only 32K of this, however, is controlled directly by the CPU.

A Little History

Texas Instruments provided a way of expanding memory back in 1980 with an add-on 32K peripheral which plugged into the side of the console. This unit was rather large and very expensive (about $250) and is virtually unobtainable these days.

About three years later TI introduced the Peripheral Expansion Box (PEB) which allowed a large range of expansion interfaces, including 32K memory, to be plugged into a mother board. Although this actually increased the cost of a memory expansion to $350 ($200 for PEB, $150 for 32K), the package now had room for disk drives, printer (RS232) card, etc, etc. With the demise of the stand-alone peripheral, 99ers had little choice but to pay up.

In 1983, when the end came for TI Home Computer Division (see John Paine's very interesting article on this subject), hundreds of thousands of consoles were bought by consumers worldwide at bargain prices. Although the cost of the computer was very attractive ($199 in Australia), the cost of fully expanding was still very high (approximately $1000). In addition, there were only a very small number of expansion systems actually produced by TI, so demand exceeded supply and second hand prices were often no better than new prices.

The situation remained this way for some time until a break through occurred in Perth. Two industrious 99ers (Bernie Elsner and Phil West) discovered that it was possible to fit the memory expansion inside the console. The availability of static RAM chips (at a reasonable price and memory capacity) simplified the project greatly as the console was able to supply the power for the chips. In contrast, the TI memory card has a total of about 28 chips and draws about 0.6 amps from an external supply.

Bernie and Phil called the project a "Matchbox Memory Expansion" as it was about the same size and shape as a match box. The expansion consisted of four memory chips soldered one on top of the other and was wired to the back of the module (GROM) port, with a small number of wires going to the mother board. Old 99ers will remember the excitement that this project caused when it was first published. This was the turning point for many users who could now "get serious" with their computer. In 1985, however, the cost of the 8K chips was $25 each (down from $140 the previous year!). Bernie predicted that the prices would ultimately drop to under $5. How right he was, although recent fluctuations in supply and demand have seen prices as high as $9 for the 8K chips.

The availability of printed circuit boards (designed by club members) has greatly simplified construction and hundreds of these add on expansions have been built by TI99/4A users.

Technical Description

The memory expansion uses 28 pin Hitachi (or equivalent) HM6264LP15 8K x 8 bit SRAM chips. (The single chip version is the HM62256.) These chips are high speed static CMOS random access memory with low operating current (60 mA) at 5 volts, and very low standby current of about 2 uA making them suitable for battery backup. SRAMs also simplify circuit construction as no memory refresh cycle circuitry is required as is the case with dynamic RAMs.

The expansion interface consists of connecting 13 address lines (A3 to A15), 8 data bus lines (D0 to D7), the Write Enable (WE), +5 volts and ground, all to the back of the GROM port. The decoding for the memory expansion is available on the TI99/4A mother board where five additional wires are connected. These are the 4 chip select (CS) lines (one for each bank of 8K) and the DBIN line. One 8K bank of memory is at address >2000 (low memory). The other 3 by 8K banks of memory are from A000 to >FFFF (high memory).

Do It Yourself Memory Expansion

If you want memory expansion (and you should), then doing it yourself will provide you with the most useful addition to your computer since the Extended BASIC module, which, incidentally, is necessary to be able to use the extra memory for BASIC programs.

Printed Circuit Boards

Several PCBs have been designed over the years to provide either memory expansion alone or in combination with other functions. Some use four 8K SRAMs while others use the newer 32K chips. All provide the same facility, and the choice of which way to go will depend on what you ultimately want to end up with. Another consideration will be the cost of the 8K versus the 32K chips. Currently one 32K chip is a little cheaper than four 8K chips.

Two PCBs are available for the 8K chip version of the memory expansion: one designed in Sydney and the other in Wollongong. The Sydney PCB is identified by the fact that the data and address lines are at different ends of the board. See the notes at the end of this article.

Four PCBs are available for the single chip expansion system:

1) A PCB which uses one 32K chip and an additional 74LS09 control chip.

2) The Peter Schubert designed Multifunction card (MFC). The MFC is a PEB card and is potentially the most powerful as it also offers the ability to have a double density disk controller, an RS232 and a PIO as well as the memory expansion option. This card can also be used with the Direct I/O Interface system described in recent issues of the TND.

3) The Mini PE System mother board (also a Peter Schubert design) can accommodate a 32K memory chip.

4) The USA designed Zeno Board. This PCB will also allow Extended BASIC, Speech Synthesizer and a number of additional GROM chips as well as a 32K memory chip. The Zeno board must be ordered from the USA and I hope to have one for show in the near future.

General Construction Hints

SRAM chips are easily damaged by static electricity. Ensure that you earth yourself to a metallic object (which is in itself earthed), prior to touching any SRAMs. Avoid touching the pins when handling the chips.

All PCBs should be inspected, prior to construction, for breaks or shorts. etc.

Use sockets for all chips. This will facilitate trouble shooting later if needed.

Inspect the module port for wear prior to starting. If OK, wash it out with alcohol prior to soldering. If it is worn, throw it away as it may cause lock up problems later. The club has a number of GROM port PCBs for replacing the old ones.

Use a soldering iron with a fine (<1.5 mm) tip, as well as fine resin-cored solder.

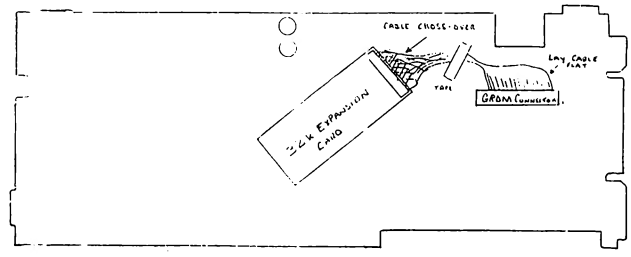
Kits - Illawarra Version PCB

32K memory expansion kits will be available through the club. See me (Lou) or the shop for supplies. Note that this version of the PCB has all of the I/O connections located at one end of the board.

PCB Location

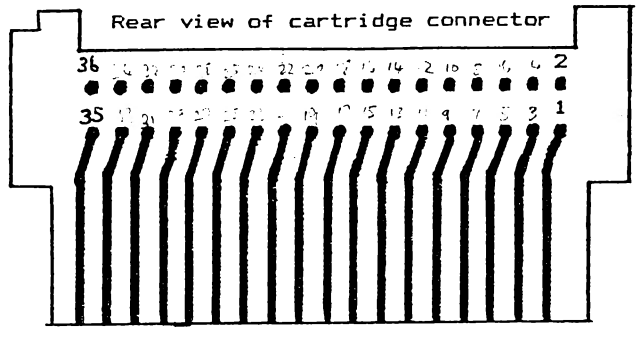
Some of the diagrams presented below are courtesy of TIUP, Perth WA.

The internal memory expansion is located on top of the mother board metal shield and close to the GROM port. The board is fixed with double sided tape between two of the memory chips and the metal shield. The distance between the PCB and the GROM port should be no less than about 50 mm. The connecting wires or cable must be laid flat to allow proper fit of the console plastic top during re-assembly.



The GROM Port Connections

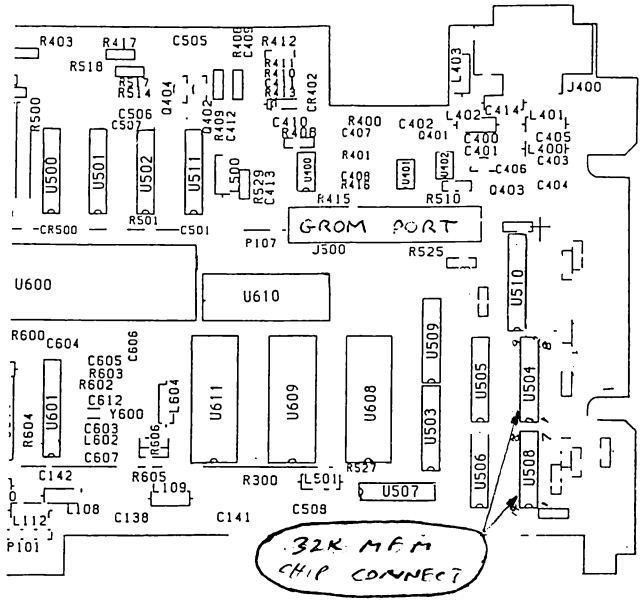
All internal versions of the memory expansion are hard wired directly to the back of the GROM port.



Looking at the back of the port, the pins are numbered from right to left as indicated above. Fine multicored wire or ribbon cable at least 200 mm long is required. Strip back and pre-tin all connections. If you are using a ribbon cable, solder every second wire to the top row of pins on the GROM port.

The Mother board Connections

Five wires must be connected from the expansion board to two chips on the TI mother board . The chips are U504 and U508 as indicated on the mother board layout below:



The pin numbers of the mother board chips are marked on the diagram.

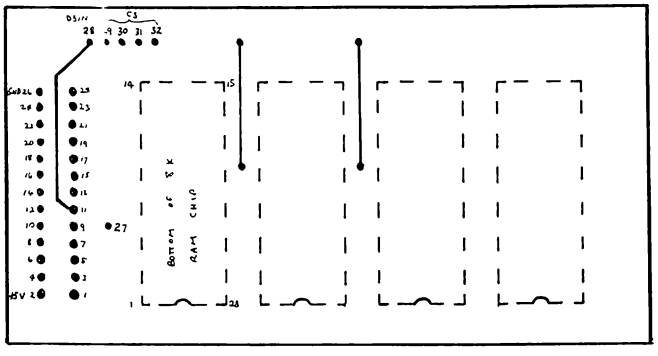
Use fine multicored coloured wire at least 300 mm long and route the wires past the GROM port and through the opening on the metal shield at the back of the console.

Connect pin 9 of U508 to DBIN (28) on the memory expansion PCB.

Connect pins 7, 9, 10 and 14 of U504 to chip select pins 29, 30, 31 and 32 respectively on the memory expansion PCB

Mounting The PCB Components

The board is easily constructed using 3 wire links, four 28 pin IC sockets and a 10 uF TAG electrolytic capacitor. The capacitor is soldered between the +5 volts and ground at the opposite end of the board to the I/O connections (watch polarity). Solder the IC sockets with the notch facing as indicated. Note that the diagram shows the bottom of the PCB.



Illawarra version of the four by 8K, 32K memory expansion showing I/O solder pad identification. The board is mounted upside down prior to soldering the connections.

With the GROM port prewired as indicated above, insert the GROM connector into the mother board socket, locate the expansion board (upside down) as indicated above and proceed to wire the ribbon cable directly to the contacts from the solder side of the board.

Using the table below for reference, strip, pre-tin and solder each wire in turn to the correct PCB pad on the expansion board. Ensure that the cable or wires can be laid flat for at least 50 mm from the GROM port.

Connections For The Illawarra PCB

GROM Port	32K Pad	Function
1	N/C	N/A
2	25,26	GND
3	16	D7
4	N/C	N/A
5	18	D6
6	N/C	N/A
7	20	D5
8	17	A15
9	22	D4
10	14	A13
11	24	D3
12	12	A12
13	23	D2
14	10	A11
15	21	D1
16	9	A10
17	19	D0
18	8	A9
19	1,2	+5 V
20	6	A8
21	N/C	N/A

GROM Port	32K Pad	Function
22	5	A7
23	15	A14
24	3	A3
25	N/C	N/A
26	7	A6
27	N/C	N/A
28	13	A5
29	N/C	N/A
30	27	A4
31	N/C	N/A
32	4	WE
33	N/C	N/A
34	N/C	N/A
35	25	GND
36	26	GND
U508 p9	28	DBIN
U504 p7	29	CSE
U504 p9	30	CSC
U504 p10	31	CSA
U504 p14	32	CS2

Testing The Memory Expansion

When you have finished wiring the memory expansion, check it all again for possible errors. If you are happy that all is well, partially assemble the console, plug in the GROM port with Extended BASIC attached, power up the console and type in the command "SIZE". If all is well you should be greeted with the following message:

13928 BYTES OF STACK FREE
24488 BYTES OF PROGRAM SPACE FREE

If not, and/or the computer locks up, then switch off the power and check for errors. In particular, make sure that there are no shorts between the solder pads and adjacent copper tracks.

Re-assemble the console carefully and start to enjoy the advantages of a highly versatile expanded TI99/4A.

Additional Notes

The following notes were written by Ross Mudie for the TIsHUG memory expansion workshops which were held in the past. The notes provide a good reference to dismantling the console as well as other construction hints. All references to PCB layouts and wiring connections in this article are for the Sydney version of the PCB (no longer readily available).

Coming Up Next Issue

In the next issue of the TND I will describe how to build the single chip versions of the memory expansion and the advantages that each system offers.

32K Memory Expansion

Inside the console

by Ross Mudie

DISCLAIMER.

The author and TIsHUG will accept no responsibility or liability for any damage to any computer as a result of use of any information contained herein.

OBJECT.

This document describes a method of installation of a 32K Static RAM printed circuit board. The board is mounted on top of the PCB shield with 2 layers of double sided foam mounting tape. Three ribbon cables connect to the PCB, two connecting to the rear of the GROM port socket and one (a long one) connects to pins of Integrated circuits U504 U508 on the main PCB of the computer. This cable is left long enough to allow the computer to be easily serviced.

MATERIALS.

Quantity	Size	Description or Use
1	10 way by 300mm	Ribbon Cable, +5V, DO-D7 and Earth.
1	14 way by 250mm	Ribbon cable, 'notWE', (*WE) and A3 to A15.
1	5 way by 500mm	Ribbon cable 'not DBIN', (*DBIN) and Chip select wires CS2, CSA, CSC and CSE.
4	45mm by 12mm	Double sided "foam type" mounting tape. (PCB mount).
1	20mm by 12mm	Double sided "foam type" mounting tape. (Retain ribbon)
1		Memory Expansion Printed Circuit board.
1	22uF 10V	Tag Tantalum capacitor.
4	28 pin	DIL IC sockets.
4	HM6264LP-15	CMOS Static RAM.
1	Metre 0.7mm	flux cored solder. (Only 0.5 metre should be needed.)

A small amount Solder flux solvent.

WARNING...

Do not remove the CMOS RAM chips from the anti-static packaging until you have completed installation of the PCB in the console, then only under anti-static conditions. Only work on the console under anti-static conditions.

TOOLS REQUIRED.

(a) Soldering Iron, preferably temperature controlled, otherwise not exceeding 20 watts, with a maximum soldering face of 1.5mm.
(b) Fine wire cutters.
(c) Fine long nosed pliers or wiring tweezers.
(d) Medium sized phillips head screw driver.
(e) Wire stripper.
(f) Solderwick or equivalent solder removing braid, size 1 (smallest).
(g) Anti-static work mat, wrist, job and earth straps.
(h) A small shifting spanner.
(i) A short stiff bristle brush or toothbrush.

OVERVIEW OF METHOD.

This is a summary of the steps required in this modification.

(a) Read this instruction thoroughly before starting.
(b) Assemble PCB, connect the ribbon cables to the PCB, check and de-flux.
(c) Dismantle console, remove GROM connector, connect two of the ribbon cables to the GROM port connector.
(d) Remove shield from the main PCB of the computer, connect ribbon cable.
(e) Reassemble PCB shield, mount expansion memory PCB, plug in memory chips.
(f) Reassemble computer and test.

Carry out all work under anti-static conditions.

METHOD.

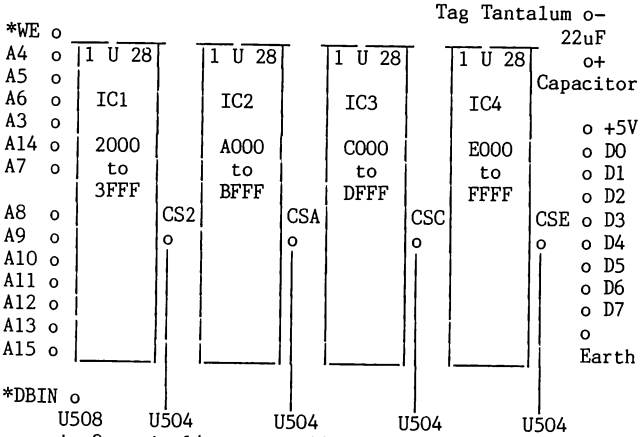
(b) Place the IC sockets in the PCB with correct orientation and solder carefully. The notch in one end of the sockets should be in line with the position for the Tag Tantalum capacitor.

Terminate the 10 way ribbon cable on the capacitor end of the PCB. Note which wires are used for the +5V and Earth. The +5V is at the end near the capacitor, the earth connection is at the other end of the group of 10 terminations, refer to Figure 1.

Terminate the 15 way ribbon cable to the group of in-line connections from *WE at the pin 1 end of the ICs to A15.

Terminate the 5 way ribbon cable to the *DBIN, CS2, CSA, CSC and CSE.

Place the 22uF Tantalum capacitor in the PCB, observing correct polarity and solder in carefully. Cut the leads off short.



Expansion RAM PCB Layout and Connections
Figure 1

Check the PCB thoroughly for solder bridges between tracks and solder pads. especially where the ribbon cable wires terminate in the PCB.

After visually ensuring that the board is free of faults clean the solder side of the board with solder flux remover and a new (cheap) toothbrush or a brush with short stiff bristles. The purpose of this is to clean off excess flux residue left from soldering. Take care to avoid flux and solvent getting into the IC socket holes.

Recheck the soldering of the board after cleaning.

When the board is dry place two double layers of foam tape on the under side of the board, leaving the protective paper covering in position on the exposed face of the tape.

The actual order that the Address wires and the Data bus wires are inter-connected with the computer does not matter, but the data bus and address must not be interchanged with each other. The wires for +5V, earth, *WE and *DBIN must be connected as shown. The connection of the CS wires may be in any order, (the computer will not mind which chip is used for each memory block), but if the diagrams herein are followed it will be connected in logical order.

(c) Set up a Static Electricity free work place with anti-static mat earthed. Remove any cartridge from the GROM port. Disconnect any cords.

Place the console upside down on an anti-static surface which will not scratch the console, with the front of the console nearest to you.

Remove the 7 screws which retain the base of the computer.

Take the power input wire from under the piece of tape, remove the power supply socket from the rear of the computer and lay the socket and wire to the side.

Remove the 3 screws which hold the main board in place. (On some consoles it will be necessary to remove the screws which retain the power supply board also.)

Ensure that anti-static wrist strap is attached.

Carefully lift up the main board and unplug the key board connector, without touching any contacts.

Invert the main board and place it on the anti-static mat nearest to you. Remove the right angle shaped GROM port connector from the main PCB unit to prevent possible static discharge damage to the computer. Do not touch the contacts of the GROM port connector and ensure that the contacts remain clean.

		* WE	A4	A5	A6	A3	A7	A8	A9	A1 0	A1 1	A1 2	A1 3	A1 5			Ear th
36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2

						A1 4		+5 V	D0	D1	D2	D3	D4	D5	D6	D7		
35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1	

Rear View of GROM Port Connector
A=Address line; D=Data bus line;
+5V = +5 volt power supply
Designations for pins connected to in this modification only are shown in this diagram.
Figure 2

In connecting the ribbon cable remember that the flat cables will need to lay flat on top of the PCB shield with the wires running individually up to the pins of the GROM port connector. The wires connect to the visible solder pins of the GROM port connector unit by placing a turn of wire around the socket pins before soldering.

Allow 45mm from the nearest edge of the PCB to the GROM connector with the Tantalum capacitor end of the PCB furthest away from the GROM connector.

Refer to Figure 2 for the GROM connector numbering.

Connect the 10 way ribbon connector to pins 2, 3, 5, 7, 9, 11, 13, 15, 17 and 19 of the GROM port connector. Start with the earth wire, (as identified on the expansion PCB), on GROM connector pin 2, then continue with the wires in turn. The last wire is the +5V which connects to GROM connector, pin 19. Connect the 15 way ribbon cable to pins 8, 10, 12, 14, 16, 18, 20, 22, 23, 24, 26, 28, 30 and 32 of the GROM port connector.

Check that there are no shorts between pins and that there is no excess bare wire on any of the connections. Watch out for individual strands of the ribbon cable wires.

(d) Ensure anti-static wrist strap is attached and PCB earth (bottom PCB shield) is contacting the anti-static mat.

Remove the top cover of the main PCB, it is held on by 3 nuts and bolts and two sheet metal clips. Note which clip came from where, they must go back in the same places or the case will not fit properly. (I also prefer to undo the two phillips head screws holding the metal case to the heatsink of the video processor as this saves having to worry about the thermal compound. GT)

Do not disturb the white thermal compound which improves the thermal conductivity of the heat away from the Video Display Processor chip.

* Ensure that the soldering iron tip is connected to the earthed anti-static mat

Remove any twists from the 5 way ribbon cable.

Leave a reasonable amount of slack in the 5 way ribbon cable and solder the other end of this cable on the main PCB IC pins as follows:

*DBIN connects to U508 pin 9
CS2 connects to U504 pin 14
CSA connects to U504 pin 10
CSC connects to U504 pin 9
CSE connects to U504 pin 7

Take special care at this point not to provide any solder bridges between adjacent pins or IC pins and PCB tracks.

Place the 5 way ribbon cable so that it passes out the opening in the shield near the cassette socket. The ribbon cable should be positioned past the corner of the GROM connector nearest the cassette socket.

Apply a small amount of fresh thermal compound to the VDP chip if the existing compound has dried out.

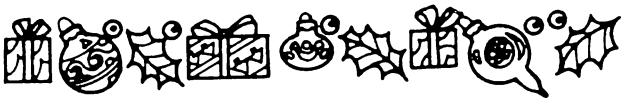
Replace the top cover of the main PCB shield, ensure that the clips hold correctly, especially the one near the VDP chip (the one with the white thermal compound on top). It may be necessary to squeeze the clips a bit, whilst they are off, so that they hold tight. (If you removed the screws into the video processor heat sink, then replace them here. GT)

Ensure that the edge connector of the GROM port connector is clean, then plug the GROM connector into the main board of the computer.

Remove the paper covering from the foam tape and locate the PCB on top of the shield approximately 45mm to the inside of the computer from the GROM connector, so that the air circulation holes are not blocked. Correct placement of the PCB will ensure that is does not foul the top cover of the console case. Watch that the ribbon cables do not lie across the mounting holes, there are two holes which the ribbons may lay across, possibly use a piece of double sided mount to hold the ribbons in position here, avoiding the mounting holes.

Under anti-static conditions remove the memory chips from their packaging and plug them into their sockets; correct orientation is essential. The end with the notch goes goes to the same side of the board as the Tag Tantalum capacitor. It may be necessary to bend the pins of the ICs in a little so that they will plug in easily into the sockets. This is best done by placing the whole of one side of pins on the flat surface of the table and then by applying even pressure bending all pins on one side in a uniform manner. Both sides should receive equal treatment. When plugging the ICs into the sockets ensure that no pin bends back or folds up underneath the IC.

Reassemble the computer. Plug in Extended BASIC and on entering Extended BASIC type SIZE <ENTER>. The computer should have 13928 bytes of STACK FREE and 24488 bytes of PROGRAM SPACE FREE. o



continued from page 26

```
WAIT 5
CLEAR
DISPLAY STATUS
RETURN
*
* FIN1       Save as  FIN1/C
* ****  Finish Program 11/10/88
*
 ****************************
```

We have previously covered everything in this system. There are a couple of tricks that might be of interest. In INFSCR1 the WHILE loop will not accept a value less than zero or greater than 5, and in PR-LBLS1 there is a WHILE within a WHILE, and they both use the same variable (ANS). I have not typed in complete command files. I used FunnelWeb to edit command files we covered earlier, and I deleted and added some lines and then saved them under a new name. Read over the old tutorials. You will find it all there, with explanations. I searched for NM in the label command file, because it is the easiest and least confusing for the user. We created the NM field in tutorial 4. In January 1989 I plan on starting with the new features of TI-Base Version 2.0. I am sure I will recover some of this old stuff in an effort to compare version 1.02 with version 2.0. If you are lost, do not despair. Keep those questions and tips coming in. And, I would like to thank Jerry Keisler of the PARIS 99/4A User Group. He gave me several good tips. Thanks Jerry. o

Extended BASIC Screen Colours

Author unknown
(Extended BASIC plus 32K RAM required)

Here is a super short, super fast, assembly routine for Extended BASIC that allows you to change screen and character colours instantaneously!

There are lots of possible uses for the thing including games, but the real feature of this program is that it changes the colour of the edit Mode screen as well! Yes, no more black on cyan if you do not want to!

How does it Work?

The colour change is inserted into the user defined interrupt and is constantly "re-performed" every 1/50 of a second. This makes it seem like the edit Mode screen colour has been changed. In order to return control of the colour commands CALL SCREEN and CALL COLOR, you must load the user defined interrupt with zeros (eg CALL LOAD(-31804,0,0). Any use of CALL COLOR or CALL SCREEN while the routine is operational will just cause the screen to flash briefly.

Demonstration Programs

Along with the program that loads in the original routine, below is a demonstration routine to show off your new screen colours.

A simple CALL LOAD will do it.

Of course, you do not need a program to change to screen colours once the original file is loaded. All you have to do is poke a single byte value into CPU address 9460. This value is found by doing the following:

Foreground colour (0-15)x16 plus
Background colour (0-15)

For instance, to set the screen to black and the characters to white you would do the following:

15x16+1=241
CALL LOAD(9460,241)

NOTE: 0=transparent, 15=white

PROGRAM #1: SCRNCOLR/X

```
100 ! ****************
110 ! *              *
120 ! * SCREEN COLOR *
130 ! *              *
140 ! ****************
150 !
160 !      11/84
170 !
180 !    SUBFILE99
190 !
200 CALL CLEAR :: CALL INIT
210 MEM=9459
220 !
230 ! *LOAD IN PROGRAM*
240 !
250 FOR I=1 TO 50
260 READ X
261 TOT=TOT+X
270 CALL LOAD(MEM+I,X)
280 NEXT I
285 IF TOT<>3951 THEN 420
290 !
300 ! *START UP PROGRAM*
310 !
320 CALL LOAD(8194,37,38,"",-31804,36,246)
330 END
340 !
350 ! *PROGRAM DATA*
360 !
```


```
370 DATA 244,0,2,1,0,135,208,96,36,244,216
380 DATA 1,140,2,6,193,216,1,140,2,2,1,0,72
390 DATA 216,1,140,2,6,193,216,1,140,2,2
400 DATA 0,0,32,216,32,36,244,140,0,6,0
410 DATA 22,251,4,91
420 PRINT "DATA LINES ENTERED WRONG.   CHECK AGAINST
    LISTING!"
430 END
```

PROGRAM #2: CLRDEMO1/X

```
100 ! ****************
110 ! *              *
120 ! *  COLOR CHANGE *
130 ! *              *
140 ! * DEMONSTRATION *
150 ! *              *
160 ! ****************
170 !
180 !       11/84
190 !
200 !     SUBFILE99
210 !
220 ! *---------------*
230 ! NOTE:
240 !     YOU MUST HAVE
250 ! ALREADY LOADED AND
260 ! RUN "SCRNCOLR/X"!
270 ! *---------------*
280 !
290 CALL CLEAR :: RANDOMIZE :: DIM C$(15)
300 M$="Screen Colour Change Demonstration "
310 !
320 FOR C=0 TO 15 :: READ C$ (C):: NEXT C
330 !
340 !
350 DISPLAY AT(1,1):M$::DISPLAY AT(2,1):
    RPT$("-",LEN(M$))
355 DISPLAY AT(10,1):"FRGRND COLOUR:"
356 DISPLAY AT(16,1):"BKGRND COLOUR:"
360 !
370 FC=INT(15*RND)
380 BC=INT(15*RND)
390 DISPLAY AT(10,14):C$(FC)
400 DISPLAY AT(16,14):C$(BC)
410 !
420 CVAL=16*FC+BC
430 CALL LOAD(9460,CVAL)
440 !
450 GOTO 370
460 !
470 DATA Transparent,Black,Medium Green,Light Green
480 DATA Dark Blue,Light Blue,Dark Red,Cyan
490 DATA Medium Red,Light Red,Dark Yellow,Light Yellow
500 DATA Dark Green,Magenta,Gray,White
510 END
```

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : CO-ORD

Modules Wanted:
Early Learning Modules...three
Addition and Subtraction...three
Connect Four...two
Addition 5 to 6 years ....one
Other modules with games for very young children.
Joysticks and adaptor if necessary.

If you can help, then please ring the International Kindergarten at Chatswood. Ask for Mrs Sylvia Kwok on (02)419 3499.
Dick Warburton.

MAIL TO : ALL
MAIL FROM : CHEMTECH

Can anybody recommend a good quality, robust joystick. Re-wiring to suit the TI is no problem. Thanks and regards.....Tony Beuermann.

Games Information

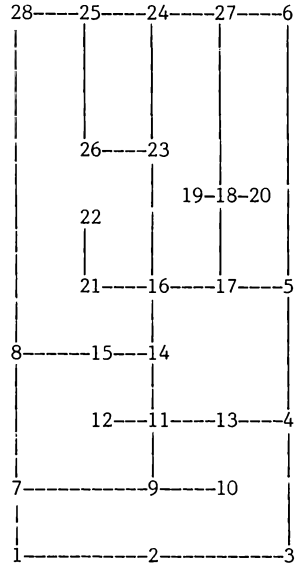
by Robert Brown

Welcome to yet another NEW LOOK GAMES INFO. The date is in the middle of the holidays, and as my exams are approaching and I am doing Year 12, which means HSC next year, this will most likely the last set of articles for some period of time. I hope to write a few articles at this time and get them to the editor each month. So enjoy it while it lasts.

This month, we have the long waited Karate Challenge Review, as well as the solution for the Infocom Adventure Witness.

Witness Solution

1 GO TO #8, AND RING THE BELL
2 WAIT FOR PHONG TO LEAD YOU TO #24
3 ASK MONICA ABOUT MR. LINDER
4 WAIT UNTIL LINDER FINISHES HIS DRINK
5 SIT ON THE WOODEN CHAIR WHEN YOU GET TO THE OFFICE.
6 STAY THERE UNTIL LINDER IS KILLED
7 PUSH THE OFFICE BUTTON
8 ASK PHONG ABOUT THE OFFICE BUTTON TO
9 MAKE HIM ADMIT TO THE CONSPIRACY
10 LOOK AT THE CLOCK
11 LOOK AT THE KEYHOLE
12 GET THE POWDER.
13 GO TO #26 TO GET PHONG
14 GET THE HOUSE KEYS FROM HIM
15 GO TO #21 AND OPEN THE BOOK
16 READ THE RECEIPT
17 GO TO #13
18 UNLOCK AND OPEN THE BACK DOOR
19 GO E
20 MAKE A CAST OF THE FOOT-PRINTS IN #4
21 GO S
22 ANALYZE THE GUN IN #3
23 GO TO #24 AND COMPARE THE CAST TO STILES' SHOES
24 GO TO #10 AND CONFRONT MONICA RIGHT AFTER SHE RETURNS
25 SHOW HER THE POWDER TO MAKE HER ADMIT TO THE CONSPIRACY
26 ACCUSE HER
27 GO TO #17 AND LOOK AT THE DESK CAREFULLY
28 READ THE REPORT
29 ASK MONICA ABOUT IT WHEN SHE GOES TO HER ROOM (#17) WAIT UNTIL SHE GOES OUT OF HER ROOM AGAIN, AND THEN FOLLOW HER TO #13
30 HANDCUFF HER
31 SEARCH HER TWO TIMES
32 ASK HER ABOUT THE HIDDEN HAND GUN
33 ASK HER ABOUT THE CLOCK KEY
34 ARREST HER



LAYOUT OF THE WITNESS

Karate Challenge

To my knowledge, I was the first person in the club to actually play this game. If you wish to purchase it, you can get it from Triton for $14.95 (plus tax). But you need 32K as well as a disk drive.

At first glance, the game seems simple enough, two figures fighting with scorre and power arrows on the screen.

The key codes are given at the beginning of the game; you get 3 punches, 3 kicks and the ability to move left or right. All the blows have a different travel length, with the kicks reaching out much further. The high and the low punches can also be used as blocks, so that means you have a ring of defence.

Power arrows regenerate when you do not get hit for a few seconds, same for the other side, up to a maximum of 15. There are 26 levels (A to Z), and you fight students up to level J. There is a dragon on level J and could also be more dragons on your way from J to Z. After level N, you run out of students and get masters. There are point systems and arrow bonuses, but that is a long way off.

I highly recommend you buy this game, it is excellent.

Well that just about finishes this NEW LOOK ARTICLE. Do not forget that you can write to myself at...

141 Beecroft Road, Beecroft 2119 or

write to GAMES on TEXPAC.

Bye for now... o

continued from page 6

The pre-scan switch does not have much effect unless the program is of substantial size, so it is not worth worrying about too much in the early stages of a program's development beyond being prepared for the possibility. The Extended BASIC manual supplement (p10) shows how all variable and sub-program declarations may be gathered together to minimize the range of the pre-scan, by using a GOTO to jump over the list to the first executable statement. This can be gotten away with since Extended BASIC does not do a complete check for correct syntax until it comes to execute the line. This is the only virtue one can ascribe to Extended BASIC's failure to reject all invalid lines at entry time. The same technique can be used within a sub-program, and I have found it very convenient for this same GOTO to reserve a hiding place in which to tuck away the subroutines accessed by GOSUBs within the sub-program. o

continued from page 2

I hope the club is interested in my system, but if not, would you please run an advertisement for the lot at $250. My phone number is (058)81 3959 and I am usually home after 4pm.

Yours sincerely, Greg J Bull

PS All modules have the original manuals and I have the following books:
1. Tantalising Games for your TI99/4A; Hel Renko/Sam Edwards
2. The Best of TI99/4A Cartridges; Thomas Blackador
3. Video Display Processors TMS9918A/TMS9928A/TMS9929A Data Manual
4. Video Display Processors Programmers' Guide
5. TMS32010 Development Support
6. TI99/4A Users Reference Guide x2
7. 32 BASIC Programs for the TI99/4A; Rugg, Feldman and Allen o

Project Management

Author unknown, supplied by John Paine

In the Beginning the Project Manager created the Programming Staff. The Programming Staff was without form and Structure. And the Project Manager said, "Let there be Organization"; and there was Organization. And the Project Manager saw that Organization was good; and the Project Manager separated the workers from the supervisors, and he called the supervisors "Management", and he called the workers "Exempt".

And the Project Manager said, "Let there be a mission in the midst of the Organization, and let it separate the workers, one from another". And the Project Manager created the mission and he called it "The System". And the Project Manager separated those who were to benefit from the System from those who were to build it. And he called the former "User", and he called the latter "Programmers".

And the Project Manager said, "Let all the Programmers in the Organization be gathered together into one place, and let a Chief Programmer be brought up to lead them". And it was so. And the Project Manager saw that it would work.

The Project Manager said unto the Chief Programmer, "Create for me a schedule, so that I may look upon the schedule and know the Due Date". And the Chief Programmer went among his staff and consulted with them. And the staff was divided into two parts, one part was called "Analysts", and the other part was called "Application Programmers". And the Analysts went back to their desks and estimated, as was their custom. And it came to pass that each Analyst brought his estimate to the Chief Programmer whereupon he collected them, summarized them, and drew a Pert Chart.

And the Chief Programmer went unto the Project Manager and presented to him the estimate saying, "It shall take ten months". And the Project Manager was not pleased and said, "I have brought you up from the depths of the staff; and you have not grasped the "Big Picture". And the Project Manager hired consultants, and authorized overtime, and he said to the Chief Programmer, "Behold and see all that I have done! The Due Date will be in five months". The Chief Programmer was much impressed and went from before the Project Manager and proceeded to implement the System.

And the Chief Programmer sent his Analysts to the users and said, "Let Specifications be written". And there were meetings, and lunches, and telephone calls. And the Specifications were written. And there was a Payday and the Happy Hour, one month.

And the Chief Programmer examined the Specifications and saw that they were too ambitious. And he separated the mandatory features from the optional features. And he called the mandatory features "Requirements," and he called the optional features "Deferred," and the Users called him names. And the Chief Programmer gave the Specifications to the Analysts and said, "Let the Requirements be Analyzed and let the Files be Designed". And it was so. And the Chief Programmer said, "Let the Software Houses put forth their salesmen, and let us have a Data Management System". And it was so. The software houses brought forth all manner of Salesmen who presented their packages, and claimed wondrous things for them, each according to his own file structure. And it came to pass that a Data Management System was selected; and the Chief Programmer saw that it was good. And there was a Payday and the Happy Hour, a second month.

And the Chief Programmer said, "Let there be Progress Reports, so we can monitor and control;" and there were Progress Reports. And the Chief Programmer looked upon the Progress Reports and saw that the Due Date was not to be met. And the Chief Programmer arose, pressed his suit, shaved his beard, and went unto the Project Manager and grovelled. And the Chief Programmer pointed his finger, and caused blame to issue forth upon all manner of creatures who sold Hardware and Software. And the Chief Programmer asked for an Extension.

And the Project Manager was exceedingly angry, and cast doubts upon the Chief Programmers ancestry; and uttered a multitude of threats. But it came to pass that an extension was granted; and the Chief Programmer took the extension back to the programming teams, and there was much rejoicing. And the programming of the modules was completed. And there was a Payday and the Happy Hour a fifth month.

And the Chief Programmer said, "Let the modules be integrated one with another, so that System Testing may begin". And it was so.

Great difficulties were experienced, and many hours of overtime were used, and many cups of coffee were consumed. And it came to pass that System Testing was completed. And there was a Payday and the Happy Hour, a sixth month.

Then the Chief Programmer did go to the Project Manager and said unto him, "Behold, I bring you good tidings of a great joy which will come to all the Users; for on this day The System is completed". And suddenly there was with them a multitude of Users praising the Chief Programmer and saying, "Glory be to The System in the highest, but can you make this one small change?". o

continued from page 27

The Complete Forth, by Alan Winfield (Wiley)

This book has several pluses which make it a good text on Forth. First, Mr. Winfield has packed it with a generous amount of examples. He included several examples showing equivalent program segments in BASIC and Forth. He also devoted a large part of his book to the handling of INPUT and OUTPUT of both numbers and strings including an excellent development of a string variable. Finally, I like the inclusion of a Forth programmers reference card in the back of the book.

Discover Forth, by Thom Hogan (Osborne/McGraw-Hill)

This book's strong points are its appendices which have a lot of information in them and the examples Mr. Hogan uses to get his points across.

There are two additional books that I recommend for further reading by the advanced programmers among our readers.

Thinking Forth, by Leo Brodie (Prentice-Hall)

This is Mr. Brodie's second book on Forth and continues the work he started in his first book "Starting Forth". This book is devoted to Forth style and problem analysis and solving. Again it is full of tips and still more cartoons.

Threaded Interpreted Languages, by R.G. Loeliger (Byte Books)

This is a book for the experienced computer hacker who wants to learn how Forth works and how to write their own fast and compact version of a Forth-like language.

I would like to close this month's article by stating that any questions sent to me through this newsletter will be answered in one of two methods. Either (1) through the tutorial articles if it is of general interest or (2) directly if the question also includes a self addressed, stamped envelope. Please forward all your questions to the newsletter address. o

TI BASIC Program

WONKAPILLAR is written by Mark Sumner, and was commercially released in the UK by Stainless Software. It is one program I do enjoy playing. Key this in and see how you like it! The caterpillar can plant one time bomb at a time! He must not collide with himself or the walls. It is bad luck to blow your head up!!!

(Do not worry too much about your body, losing weight may be good for you!).

The length of the fuse is determined by how long you hold the space bar down. Move with usual arrow keys. In essence, to escape off the screen you must blow up lots of walls! The bomb will remove the square it is on and each of the 8 squares around it (that is a 3x3 block). Can be played in TI BASIC only. Disk owners can play from Extended BASIC if they add the utility VDPUTIL.

```
1 REM WONKAPILLAR/TI BASIC
100 U=69
110 R=68
120 D=88
130 L=83
140 B=32
150 MEN=3
160 SC=0
170 BON=3000
180 M=1
190 CALL CLEAR
200 CALL SCREEN(2)
210 CALL COLOR(9,5,2)
220 CALL COLOR(10,7,2)
230 CALL COLOR(11,13,2)
240 FOR A=1 TO 8
250 CALL COLOR(A,16,1)
260 NEXT A
270 F$="FFFFFFFFFFFFFFFF"
280 CALL CHAR(96,F$)
290 CALL CHAR(104,"BD7EDBDBFFBD423C")
300 CALL CHAR(112,"003C7E7E7E7E3C00")
310 CALL CHAR(120,"0204183C7E7E3C")
320 CALL COLOR(12,14,2)
330 CALL CHAR(128,"0000183C7E3C1800")
340 CALL COLOR(13,12,2)
350 CALL CHAR(136,"004020183C3C1800")
360 CALL COLOR(14,10,2)
370 CALL CHAR(144,"00020206060C38E0")
380 CALL COLOR(15,11,2)
390 GOSUB 2150
400 CALL CLEAR
405 BT=Z
410 CALL HCHAR(1,3,96,28)
420 CALL HCHAR(3,5,96,24)
430 CALL HCHAR(5,7,96,20)
440 CALL HCHAR(7,9,96,16)
450 CALL HCHAR(14,9,96,16)
460 CALL HCHAR(16,7,96,20)
470 CALL HCHAR(18,5,96,24)
480 CALL HCHAR(20,3,96,28)
490 CALL VCHAR(1,3,96,20)
500 CALL VCHAR(3,5,96,16)
510 CALL VCHAR(5,7,96,12)
520 CALL VCHAR(7,9,96,8)
530 CALL VCHAR(1,30,96,20)
540 CALL VCHAR(3,28,96,16)
550 CALL VCHAR(5,26,96,12)
560 CALL VCHAR(7,24,96,8)
570 CALL HCHAR(11,27,120)
580 CALL HCHAR(11,6,120)
590 CALL HCHAR(10,16,104)
600 CY=0
610 CX=1
620 PY=10
630 PX=16
640 REM KEY
650 BON=BON-10
660 SC=SC+1
670 IF BT>0 THEN 1320
680 CALL KEY(0,K,S)
690 IF K<>U THEN 720
700 CY=-1
710 CX=0
720 IF K<>R THEN 750
730 CY=0
740 CX=1
750 IF K<>D THEN 780
760 CY=1
770 CX=0
780 IF K<>L THEN 810
790 CX=-1
800 CY=0
810 IF K=B THEN 940
820 IF PY+CY<1 THEN 1470
830 IF PY+CY>24 THEN 1470
840 IF PX+CX>32 THEN 1470
850 IF PX+CX<1 THEN 1470
860 CALL GCHAR(PY+CY,PX+CX,G)
870 IF G<>32 THEN 1060
880 CALL HCHAR(PY+CY,PX+CX,104)
890 CALL SOUND(-50,-7,5)
900 CALL HCHAR(PY,PX,112)
910 PY=PY+CY
920 PX=PX+CX
930 GOTO 640
940 REM PLANT BOMB
950 BT=1
960 CALL KEY(0,K,S)
970 IF K<>B THEN 1020
980 BT=BT+1
990 IF BT=9 THEN 1020
1000 CALL SOUND(-50,-5,5)
1010 GOTO 960
1020 CALL HCHAR(PY,PX,BT)
1030 BY=PY
1040 BX=PX
1050 GOTO 820
1060 REM MAN LOST
1070 IF G>119 THEN 1270
1080 CALL SOUND(250,-2,5)
1090 PRINT
1100 MEN=MEN-1
1110 PRINT "MEN:";MEN
1115 BT=Z
1120 PRINT "SCORE:";SC
1130 FOR A=1 TO 1000
1140 NEXT A
1150 IF MEN<1 THEN 1170
1160 GOTO 1610
1170 PRINT "**********GAME OVER**********"
1180 IF SC<=HIS THEN 1230
1190 PRINT : :
1200 PRINT "     * NEW HIGHSCORE! *"
1210 INPUT "INPUT YOUR INITIALS: ":H$
1220 HIS=SC
1230 PRINT "PRESS ANY KEY TO RESTART"
1240 CALL KEY(0,K,S)
1250 IF S=0 THEN 1240
1260 GOTO 150
1270 CALL SOUND(-100,440,5)
1280 CALL SOUND(-100,880,5)
1290 CALL SOUND(-100,1760,5)
1300 SC=SC+50*M
1310 GOTO 880
1320 BT=BT-1
1330 CALL HCHAR(BY,BX,48+BT)
1340 CALL SOUND(-50,-1,10)
1350 IF BT>0 THEN 680
1360 REM EXPLOSION
1370 CALL SOUND(-100,110,5,-5,0)
1380 CALL HCHAR(BY-1,BX-1,32,3)
1390 CALL HCHAR(BY,BX-1,32,3)
1400 CALL HCHAR(BY+1,BX-1,32,3)
1410 IF ABS(BY-PY)>1 THEN 1460
1420 IF ABS(BX-PX)>1 THEN 1460
1430 CALL SOUND(300,-5,1)
1440 PRINT "OOPS! YOU BLEW YOURSELF UP!"
1450 GOTO 1060
1460 GOTO 640
1470 REM MAZE COMPLETED
1480 PRINT "MAZE";M;"COMPLETED!"
1490 M=M+1
1500 IF BON>0 THEN 1520
1510 BON=0
1520 PRINT "BONUS:";BON
1530 PRINT "SCORE:";SC
```

```
1540 PRINT "TOTAL:";BON+SC
1550 PRINT "MEN:";MEN
1560 SC=SC+BON
1570 FOR A=1 TO 1000
1580 NEXT A
1590 IF M<9 THEN 1610
1600 M=1
1610 ON M GOTO 400,400,1780,1780,1620,1620,1930,1930
1620 REM MAZE 2
1630 CALL CLEAR
1640 FOR X=3 TO 32 STEP 2
1650 CALL VCHAR(1,X,96,24)
1660 NEXT X
1670 CALL HCHAR(1,3,96,28)
1680 CALL HCHAR(24,3,96,28)
1690 FOR Y=9 TO 11
1700 CALL HCHAR(Y,12,32,10)
1710 NEXT Y
1720 CALL HCHAR(4,16,136)
1730 CALL HCHAR(20,16,136)
1740 CALL HCHAR(12,5,136)
1750 CALL HCHAR(12,27,136)
1760 BON=4000
1770 GOTO 590
1780 REM MAZE 2
1790 CALL CLEAR
1800 FOR Y=1 TO 23 STEP 3
1810 CALL HCHAR(Y,3,96,28)
1820 CALL HCHAR(Y+1,3,96,28)
1830 NEXT Y
1840 CALL VCHAR(1,3,96,24)
1850 CALL VCHAR(1,30,96,24)
1860 FOR Y=8 TO 13
1870 CALL HCHAR(Y,12,32,12)
1880 NEXT Y
1890 BON=3500
1900 CALL HCHAR(6,16,128)
1910 CALL HCHAR(18,16,128)
1920 GOTO 590
1930 REM MAZE 4
1940 CALL CLEAR
1950 FOR X=3 TO 15 STEP 2
1960 CALL VCHAR(1,X,96,24)
1970 NEXT X
1980 FOR X=18 TO 30 STEP 2
1990 CALL VCHAR(1,X,96,24)
2000 NEXT X
2010 CALL VCHAR(1,16,96,48)
2020 FOR Y=1 TO 24 STEP 2
2030 CALL HCHAR(Y,3,96,28)
2040 NEXT Y
2050 CALL HCHAR(10,12,32,10)
2060 CALL HCHAR(12,12,32,10)
2070 CALL HCHAR(11,12,32)
2080 CALL HCHAR(11,21,32)
2090 CALL HCHAR(4,4,144)
2100 CALL HCHAR(20,4,144)
2110 CALL HCHAR(20,29,144)
2120 CALL HCHAR(4,29,144)
2130 BON=5000
2140 GOTO 590
2150 REM TITLE SCREEN
2160 BT=0
2170 PRINT "ppppppppp"
2180 PRINT "       WpNKAPILLAR"
2190 PRINT "        p"
2200 PRINT "        ppppppppph"
2210 PRINT : : :
2220 PRINT "       BY M.C. SUMNER"
2230 PRINT :
2240 PRINT "    (C)1982 PS SOFTWARE": : :
2250 PRINT "  HIGHSCORE:";HIS
2260 PRINT "  MADE BY  : ";H$
2270 PRINT
2280 PRINT : : : :
2290 PRINT "PRESS: S TO START        "
2300 PRINT "       C TO CHANGE CONTROLS"
2310 PRINT "       H FOR HELP"
2320 PRINT "       E TO EXIT PROGRAM"
2330 CALL KEY(5,K,S)
2340 IF S=0 THEN 2330
2350 IF K=83 THEN 2410
2360 IF K=67 THEN 2420
2370 IF K=72 THEN 2800
2380 IF K=69 THEN 3070
2390 GOTO 2330
2400 GOSUB 2420
2410 RETURN
2420 REM DEFINE KEYS
2430 PRINT : : : :
2440 PRINT "WHICH KEY FOR UP?"
2450 CALL KEY(0,K,S)
2460 IF S=0 THEN 2450
2470 CALL SOUND(100,440,0)
2480 U=K
2490 PRINT "WHICH KEY FOR DOWN?"
2500 CALL KEY(0,K,S)
2510 IF S=0 THEN 2500
2520 IF K=U THEN 2500
2530 CALL SOUND(100,440,0)
2540 D=K
2550 PRINT "WHICH KEY FOR RIGHT?"
2560 CALL KEY(0,K,S)
2570 IF S=0 THEN 2560
2580 IF K=U THEN 2560
2590 IF K=D THEN 2560
2600 CALL SOUND(100,440,0)
2610 R=K
2620 PRINT "WHICH KEY FOR LEFT?"
2630 CALL KEY(0,K,S)
2640 IF S=0 THEN 2630
2650 IF K=U THEN 2630
2660 IF K=R THEN 2630
2670 IF K=D THEN 2630
2680 CALL SOUND(100,440,1)
2690 L=K
2700 PRINT "WHICH KEY FOR BOMBS?"
2710 CALL KEY(0,K,S)
2720 IF S=0 THEN 2710
2730 IF K=U THEN 2710
2740 IF K=R THEN 2710
2750 IF K=D THEN 2710
2760 IF K=L THEN 2710
2770 CALL SOUND(100,440,1)
2780 B=K
2790 RETURN
2800 REM HELP SCREEN
2810 PRINT "   * WONKAPILLAR RULES *"
2820 PRINT : : :
2830 PRINT "  WALL    => "&CHR$(96)
2840 PRINT "  SEGMENT => p"
2850 PRINT "  HEAD    => h"
2860 PRINT :
2870 PRINT "THE OBJECT OF WONKAPILLAR   IS TO GUIDE THE
    WONKAPILLAR TO ESCAPE FROM 8 MAZES.  "
2880 PRINT "AT FIRST THIS APPEARS IMPOSS-IBLE, BUT BY
    PLACING TIME  BOMBS YOU CAN BLOW YOUR WAY"
2890 PRINT "THROUGH THE MAZE WALLS AND  ESCAPE."
2900 PRINT : :
2910 PRINT "THE CONTROLS ARE FIRST SET: E=UP, D=RIGHT,
    X=DOWN,         S=LEFT, AND THE SPACE BAR"
2920 PRINT "CONTROLS DROPPING BOMBS."
2930 PRINT "(PRESS ANY KEY TO CONTINUE)"
2940 CALL KEY(0,K,S)
2950 IF S=0 THEN 2940
2960 PRINT "THE WONKAPILLAR'S HEAD CAN'T TOUCH ANY WALLS
    OR SEGMENTS OR ITS DOOMED.  YOU MUST "
2970 PRINT "ALSO BE CAREFULL IN THE     PLACEMENT OF
    YOUR BOMBS.  "
2980 PRINT : :
2990 PRINT "THE LONGER YOU HOLD DOWN THE SPACE BAR, THE
    LONGER IT   WILL TAKE FOR THE BOMB TO GO"
3000 PRINT "OFF (MAXIMUM: 8 SECONDS).   BE SURE TO ALLOW
    YOURSELF   TIME TO ESCAPE!"
3010 PRINT : : :
3020 PRINT "(PRESS ANY KEY TO RETURN)"
3030 PRINT : : : : : :
3040 CALL KEY(0,K,S)
3050 IF S=0 THEN 3040
3060 GOTO 2150
3070 END
```

Excerpts from "Getting Started with the TI99/4A"

by Stephen Shaw, England

FOR TO STEP

Note that in TI BASIC you must always use the variable name after NEXT. NEXT on its own is an error. In some early computers you were not allowed to transfer to another line once a FOR NEXT loop had been established, but with the TI99/4A you need not worry. You may leave a FOR NEXT loop before the loop has been completed.

Sample use:

```
100 FOR FREQ=110 TO 200
110 CALL SOUND(100,FREQ,0)
120 NEXT FREQ
```

FOR..TO..STEP may also be used to provide delays:

```
100 FOR DELAY=1 TO 300
110 NEXT DELAY
```

will take a little over a second to complete in TI BASIC.

INPUT

Try to use a separate INPUT for each variable. It is possible to input more than one variable, eg by using INPUT A,B but this requires the program user to input two numbers separated by a comma.

The TI form of input, INPUT "HOW MANY?":N uses a colon separator (:), most other BASICs use a semicolon (;).

VARIABLES

When you wish to refer to a number, you may use that number, or a 'label' representing the number. For instance, if we tell the computer:

```
A=2
```

then whenever the computer comes to 'A' (without other letters, that is, with spaces or brackets on either side), it will treat it as the number 2.

'A' is a variable, and can be allocated to any number. The TI99/4A may have variable names up to fifteen letters long. You may for instance use:

```
HIGHSCORE=12000
```

A variable representing a number is a numeric variable and a variable representing a letter, a word, or a group of words is called a string variable. A string variable always ends with the dollar sign:

```
MESSAGE$="YOU WIN"
```

Strings (as they are called) are dealt with later.

READ...DATA...RESTORE

TI BASIC is slow at reading DATA lines, and if you need to use a number of READs, it is essential that you do not do it more often than absolutely necessary. It is a good idea to fill a variable array, and refer to that (ARRAYs are dealt with at some length later).

For example:

```
FOR I=1 TO 5
READ A
IF A=1 THEN 200
NEXT I
DATA 2,3,1,0,6
```

if used often, could be replaced with:

```
FOR I=1 to 5
READ B(I)
NEXT I
```

then when a check is required

```
FOR I=1 TO 5
IF B(I)=1 THEN 200
NEXT I
DATA 2,3,1,0,6
```

It is worth mentioning that DATA causes more problems in debugging a program than any other command. There must be enough DATA to fill all the READs in the program, and they must be numbers if a numeric variable is READ. Be careful how many commas you use in your DATA lines. Too many or too few can cause many hours searching for errors. The error messages you will receive may be some distance from a READ line, if you have loaded an incorrect value into a numeric variable due to missing out just one comma.

DATA hint: adding an additional value to your data list, which is never read or used, causes the computer to be notably faster in reading the actually used last item.

PRINT

TI BASIC has a fairly slow screen scroll, but your information will appear more quickly if you use the print separators instead of a number of separate PRINT lines. You will also save memory.

For example:

```
100 PRINT "PRESS"
110 PRINT "1.  TO START"
120 PRINT "2.  TO TERMINATE"
130 PRINT
140 PRINT "H FOR HELP"
```

will appear more quickly if you use:

```
100 PRINT "PRESS":"1.  TO START":"2.  TO  TERMINATE"::"H
FOR HELP"
```

TI BASIC allows you to key in a program line up to 4 screen lines long, so use this facility. Notice that instead of a single PRINT to scroll one line, an extra colon has been used in our single line amendment. Each colon causes the screen to scroll once.

continued from page 23

"getc" has a special character to signify the end of a file. "EOF" represents this character, and is one of the "#define" statements in "STDIO". "EOF" will be replaced with this special character when the program is compiled. "!=" means not equal to, and so (c != EOF) is true if we are not at the end of file. If this is the case, the character that 'c' holds is printed on the screen and the loop repeated until the condition is false (i.e. we are at the end of the file).

Finally the file is closed using the "fclose" subprogram and the option of exiting the program given. Note that this program will only echo the file name given in the first argument of "fopen". If you save the 'c' code to this file name, when you run the program, the 'c' program itself will be echoed to the screen.

That concludes another information packed part of "Programming c99". Unfortunately there was insufficient space for numeric arrays this month, but I promise these will be covered next month. Do not forget to experiment, it is the only way to learn. You may wish to write a program that counts the number of characters, or even words, in a file. If you have any queries about using 'c99' write to me at the address in the above the program.

NEXT MONTH

We will examine arrays of both integers and characters, as well as the use of pointers.

Jenny's Younger Set

Dear Jenny,

First of all, Merry Christmas and a happy New Year to you and all the Younger Set.

I have two programs for you this time. They are "Greensleeves" and a New Year's program. The idea with the New Year's program is to let the "PRESS ANY KEY" message come up just before midnight. Then on midnight press <ENTER> to see the program work. Anyhow you can decide on what you prefer.

Wishing all a Merry Christmas,
Vincent Maker

```
50 CALL CLEAR
60 GOSUB 430
70 GOTO 100
80 GOTO 100
100 CALL SOUND(250,165,0)
110 CALL SOUND(500,196,0)
120 CALL SOUND(250,220,0)
130 CALL SOUND(250,247,0)
135 CALL SOUND(125,262,0)
140 CALL SOUND(250,247,0)
150 CALL SOUND(500,220,0)
160 CALL SOUND(250,175,0)
170 CALL SOUND(250,147,0)
175 CALL SOUND(125,165,0)
180 CALL SOUND(250,175,0)
190 CALL SOUND(500,196,0)
200 CALL SOUND(250,165,0)
210 CALL SOUND(250,165,0)
215 CALL SOUND(125,156,0)
220 CALL SOUND(250,165,0)
230 CALL SOUND(500,175,0)
240 CALL SOUND(125,156,0)
250 CALL SOUND(500,123,0)
260 CALL SOUND(250,165,0)
270 CALL SOUND(500,196,0)
280 CALL SOUND(250,220,0)
290 CALL SOUND(250,247,0)
295 CALL SOUND(125,262,0)
300 CALL SOUND(250,247,0)
310 CALL SOUND(500,220,0)
320 CALL SOUND(250,175,0)
330 CALL SOUND(250,147,0)
331 CALL SOUND(250,165,0)
335 CALL SOUND(125,165,0)
340 CALL SOUND(250,175,0)
350 CALL SOUND(250,196,0)
351 CALL SOUND(250,175,0)
355 CALL SOUND(125,175,0)
360 CALL SOUND(250,165,0)
370 CALL SOUND(250,156,0)
371 CALL SOUND(250,131,0)
375 CALL SOUND(125,123,0)
380 CALL SOUND(250,156,0)
390 CALL SOUND(250,156,0)
400 CALL SOUND(500,165,0)
410 CALL SOUND(250,165,0)
420 CALL SOUND(500,165,0)
429 GOTO 490
430 DISPLAY AT(3,1):"GREENSLEEVES
                    ============"
440 DISPLAY AT(5,1):"Alas my love you do me       wrong
    to cast me off dis-  courteously.  And I have
        loved you oh so long"
450 DISPLAY AT(9,1):"delighting in your company."
460 DISPLAY AT(9,2):"Greensleeves was all my joy,
    Greensleeves was my delight."
470 DISPLAY AT(12,2):"Greensleeves was my heart ofgold,
    and who but my lady   Greensleeves?"
480 RETURN
490 CALL SOUND(500,587,0)
500 CALL SOUND(250,587,0)
510 CALL SOUND(125,277,0)
520 CALL SOUND(250,247,0)
530 CALL SOUND(250,220,0)
540 CALL SOUND(250,175,0)
550 CALL SOUND(250,147,0)
560 CALL SOUND(125,165,0)
570 CALL SOUND(250,175,0)
580 CALL SOUND(500,196,0)
590 CALL SOUND(250,165,0)
600 CALL SOUND(250,165,0)
610 CALL SOUND(125,156,0)
620 CALL SOUND(250,165,0)
630 CALL SOUND(500,165,0)
640 CALL SOUND(250,156,0)
650 CALL SOUND(250,123,0)
660 CALL SOUND(500,587,0)
670 CALL SOUND(250,587,0)
680 CALL SOUND(125,277,0)
690 CALL SOUND(250,247,0)
700 CALL SOUND(500,220,0)
710 CALL SOUND(250,175,0)
720 CALL SOUND(250,147,0)
730 CALL SOUND(125,165,0)
740 CALL SOUND(250,175,0)
750 CALL SOUND(250,196,0)
770 CALL SOUND(125,175,0)
780 CALL SOUND(250,165,0)
790 CALL SOUND(125,156,0)
800 CALL SOUND(125,131,0)
810 CALL SOUND(125,156,0)
820 CALL SOUND(500,165,0)
830 CALL SOUND(500,165,0)
```

```
100 CALL MAGNIFY(2)
110 PRINT "PRESS ANY KEY OR <ENTER>    WHEN YOU ARE
    READY TO RUN   THE NEW YEAR'S PROGRAM."
120 CALL KEY(0,U,I)
130 IF I=0 THEN 120
140 CALL CLEAR
150 CALL SCREEN(1)
160 FOR T=0 TO 8
170 CALL COLOR(T,4,2)
180 NEXT T
190 CALL SPRITE(#1,49,4,40,110,0,0)
200 CALL SPRITE(#2,57,4,75,110,0,0)
210 CALL SPRITE(#3,57,4,110,110,0,0)
220 CALL SPRITE(#4,48,4,145,110,0,0)
230 DISPLAY AT(1,3):"** HAPPY NEW YEAR **"
240 DISPLAY AT(24,3):" HAVE A GOOD ONE !!"
250 CALL SPRITE(#5,42,4,70,50,5,0)
260 CALL SPRITE(#6,42,4,70,175,5,0)
270 CALL DELSPRITE(#5,#6)
280 GOTO 250
```

Dear Jenny,

This is Crocodile Jones. I would like to thank all those who have sent me letters this year. I wish them every success in completing their adventures. Also I wish you and the Younger Set a very merry Christmas.

Wishing all a merry Christmas and successful New Year,
Crocodile Jones

A big thank you to both Vincent and Crocodile for all their good wishes to me and our readers and also for all their contributions throughout the year. I know that school takes a lot of our younger members' time but I hope they can continue to push Dad off their TI99/4A and get some time to try out all the good software that is around. Programs are fun to write and make the computer do what you want it to instead of what someone else has made it do. There are a lot of different languages to try, some of them very powerful, and the more you do while you are young the easier it will be as you get older. Merry Christmas from me and have fun in the holidays! o

Tips from the Tigercub #37

by Jim Peterson, Tigercub Software, USA

156 Collingwood Ave., Columbus, OH 43213

Distributed by Tigercub Software to TI99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 130 original programs in Basic and Extended Basic, available on cassette or disk, only $3 each plus $1.50 per order for PPM. Entertainment, education, programmer's utilities.

Descriptive catalog $1, deductable from your first order.

Tips from The Tigercub, a full disk containing the complete contents of this newsletter numbers 1 through 14, 50 original programs and files, just $15 postpaid. Tips from the Tigercub volume 2, another disk full, complete contents of numbers 15 through 24, over 60 files and programs, also just $15 postpaid.

```
**********************************************
* Tips from the Tigercub volume 3 is now ready.     *
* Another 62 programs, routines, tips, tricks from  *
* numbers 25 through 32.        Also $15 postpaid.  *
* Any two Tips disks $27 or all 3 for $35 postpaid. *
**********************************************
```

Nuts & Bolts (No. 1), a full disk of 100 Extended Basic utility sub-programs in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using sub-programs , and 5 pages of documentation with an example of the use of each subprogram. All for just $19.95 postpaid.

Nuts & Bolts No. 2, another full disk of 108 utility sub-programs in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $19.95 postpaid, or both Nuts & Bolts disks for $37 postpaid.

Tigercub Full Disk Collections, just $12 postpaid! Each of these contains either 5 or 6 of my regular $3 catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am not selling public domain programs. My own programs on these disks are greatly discounted from their usual price, and the public domain is a free bonus!

```
TIGERCUB'S BEST,                      PROGRAM-TUTOR,
PROGRAMMER'S UTILITIES,                 BRAIN GAMES,
BRAIN TEASERS, BRAIN BUSTERS!,  MANEUVERING GAMES,
ACTION REFLEX AND CONCENTRATION, TWO-PLAYER GAMES,
KID'S GAMES,          MORE GAMES,          WORD GAMES,
ELEMENTARY MATH,           MIDDLE/HIGH SCHOOL MATH,
VOCABULARY AND READING,          MUSICAL EDUCATION,
KALEIDOSCOPES AND DISPLAYS
```

For descriptions of these send a dollar for my catalog!

I am going to mail out the July and August Tips at the end of June, and go fishing. Imagine, a TI99/4A publication ahead of schedule! However, in the unlikely event that anyone should send me an order, it will receive my usual one-day service.

Here is another tune for the dulcimer player in the last Tips. Change the TO value to 94 -

```
350 DATA 9,11,13,13,13,13,13,16,16,13,13,11,11,11,11,
    11,11
360 DATA 16,18,14,21,18,18,16,13,9,11,9,9,9,9,9
370 DATA 21,20,18,18,16,13,16,16,9,11,13,11,13,14,13,13
380 DATA 21,20,18,18,16,13,16,16,9,13,11,9,8,6,4,4
390 DATA 9,11,13,13,13,13,13,16,16,13,13,11,11,11,11,11
400 DATA 16,18,14,21,18,18,16,13,9,11,9,9,9,9,9
```

Here is one for those who like graphics, and those who make a living designing floor tiles. It borrows a bit from a Renko and Edwards program -

```
100 CALL CLEAR :: F=2 :: BC=16 :: RANDOMIZE ::
    DISPLAY AT(2,10):"ESCHER ART": :TAB(14);"by":
    :TAB(9);"Jim Peterson"
110 DISPLAY AT(12,3):"Press Q for new pattern":" R to
    change colours":" C for new colours": :" Any key to
    start"
120 CALL KEY(0,K,S):: IF S=0 THEN 120 ELSE CALL CLEAR
130 DATA 8080808080808080FF0000000000000001010101010101
    0100000000000000FF
140 DATA 2020202020202020000FF0000000000040404040404040
    400000000000FF0000
150 DATA 1010101010101010000000FF0000000008080808080808
    0800000000FF000000
160 DATA C0C0C0C0C0C0C0C0FFFF0000000000000030303030303030
    3000000000000FFFF
170 DATA F0F0F0F0F0F0F0F0FFFFFFFF000000000F0F0F0F0F0F0F0
    F00000000FFFFFFFF
180 DATA 8040201008040201010204081020408080402010080402
    0101020408102040 80
190 DATA 101020C0000000008080403000000000000000000304080
    8000000000C0201010
200 DATA FFFEFCF8F0E0C080FF7F3F1F0F0703010103070F1F3F7F
    FF80C0E0F0F8FCFEFF
210 DATA F0F0F0F0000000000F0F0F0F0F0000000000000000F0F0F
    0F00000000F0F0F0F0
220 DATA 80C0A090888482FFFF82848890A0C080FF412111090503
    0101030509112141FF
230 DATA 8142241818244281814224181824428181422418182442
    818142241818244281
240 DATA 08080808FF08080810101010FF10101010101 0FF101010
    10080808FF08080808
250 DATA AA55AA55AA55AA5555AA55AA55AA55AAAA55AA55AA55AA
    5555AA55AA55AA55AA
260 DATA F0F0F0F00F0F0F0F0F0F0F0FF0F0F0F0F0F0F0F00F0F0F
    0F0F0F0F0FF0F0F0F0
270 CALL CHAR(84,RPT$("0",64)):: FOR CH=88 TO 140 STEP 4
    :: READ CH$ :: CALL CHAR(CH,CH$):: NEXT CH :: CALL
    SCREEN(5)
280 A=INT(6*RND+3):: H=INT(24/A):: HC=INT(28/A)::
    W=ABS(HC/2=INT(HC/2)):: DIM M(8,8) :: FOR P=1 TO A
290 D(P)=4*INT(15*RND+21)
300 NEXT P :: GOSUB 370
310 CALL KEY(3,K,ST):: IF K<>81 THEN 330
320 CALL SOUND(50,500,5):: FOR J=1 TO 4 :: FOR JJ=1 TO A
    :: M$(J,JJ)="" :: NEXT JJ : : NEXT J :: GOTO 280
330 IF K<>67 THEN 360 :: F=INT(15*RND+2)
340 BC=INT(15*RND+2):: IF BC=F THEN 340
350 FOR S=7 TO 14 :: CALL COLOR(S,F,BC):: NEXT S :: GOTO
    310
360 IF K<>ASC("R")THEN 310 :: T=F :: F=BC :: BC=T ::
    GOTO 350
370 ON A-2 GOSUB 380,390,400,410,420,430 :: GOTO 520
380 RESTORE 440 :: RETURN
390 RESTORE 450 :: RETURN
400 RESTORE 460 :: RETURN
410 RESTORE 470 :: RETURN
420 RESTORE 480 :: RETURN
430 RESTORE 500 :: RETURN
440 DATA 1,2,1,2,3,2,3,1,3
450 DATA 1,2,2,1,2,3,3,2,3,4,4,3,4,1,1,4
460 DATA 1,2,3,1,2,2,3,4,3,2,3,4,5,4,3,4,5,1,5,4,5,1,2,
    1,5
470 DATA 1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3,4,5,6,6,5,
    4,5,6,1,1,6,5,6,1,2,2,1,6
480 DATA 1,2,3,4,3,2,1,2,3,4,5,4,3,2,3,4,5,6,5,4,3,4,5,
    6,7,6,5,4
490 DATA 5,6,7,1,7,6,5,6,7,1,2,1,7,6,7,1,2,3,2,1,7
500 DATA 1,2,3,4,4,3,2,1,2,3,4,5,5,4,3,2,3,4,5,6,6,5,4,
    3,4,5,6,7,7,6,5,4
510 DATA 5,6,7,8,8,7,6,5,6,7,8,1,1,8,7,6,7,8,1,2,2,1,8,
    7,8,1,2,3,3,2,1,8
520 FOR J=1 TO A :: FOR JJ=1 TO A :: READ M(J,JJ)::
    NEXT JJ :: NEXT J
530 X=A+1 :: FOR J=1 TO A :: FOR JJ=1 TO A ::
    M$(1,J)=M$(1,J)&CHR$(D(M(J,JJ)))
540 M$(2,J)=M$(2,J)&CHR$(D(M(JJ,X-J))+1)
550 M$(3,J)=M$(3,J)&CHR$(D(M(X-J,X-JJ))+2)
560 M$(4,J)=M$(4,J)&CHR$(D(M(X-JJ,J))+3)
570 NEXT JJ :: NEXT J
580 CALL CLEAR :: FOR R=1 TO A*H STEP A :: FOR C=1 TO
    A*HC STEP A
590 CALL KEY(0,K,ST):: IF K=81 THEN 320
600 V=V+1+4*(V=4) :: FOR T=1 TO A ::
    DISPLAY AT(R-1+T,C):M$(V,T):: NEXT T :: NEXT C ::
    V=V+W+4*(V=4) :: NEXT R
610 RETURN
```

This routine will search a disk file for up to 10 key words in one pass (more if you DIM K$( )) and you may elect to find all records which contain the key word or only those which contain it in combination with one of 1 or more secondary key words.

```
100 CALL CLEAR
110 Y=0 :: DISPLAY AT(3,5):"TIGERCUB KEYSEARCH" ::
    DISPLAY AT(6,1):"Filename? DSK" :: ACCEPT
    AT(6,14)BEEP:F$ :: OPEN #1:"DSK"&F$,INPUT
120 DISPLAY AT(8,1):"Output to":" (1)Screen":"
    (2)Printer":" (3)Both" :: ACCEPT
    AT(8,11)VALIDATE("123")SIZE(1)BEEP:Q
130 IF Q>1 THEN DISPLAY AT(13,1):"Printer name?" ::
    ACCEPT AT(13,15):P$ :: OPEN #2:P$
140 DISPLAY AT(15,1):"Searchfor":" (1)First match":"
    (2)All matches" :: ACCEPT
    AT(15,13)VALIDATE("12")SIZE(1)BEEP:S
150 DISPLAY AT(12,1)ERASE ALL:"Press ENTER when all
    key-":"words have been entered."
160 DISPLAY AT(17,1):"Press ENTER if none -"
170 Y=Y+1 :: DISPLAY AT(15,1):"Keyword? ";CHR$(127)::
    ACCEPT AT(15,10)SIZE(-28)BEEP:K$(Y):: IF
    K$(Y)=CHR$(127)THEN 190
180 W=W+1 :: DISPLAY AT(19,1):"With? ";CHR$(127)::
    ACCEPT AT(19,7)SIZE(-21)BEEP:W$(Y,W):: IF
    W$(Y,W)=CHR$(127)THEN W=0 :: GOTO 170 ELSE GOTO 180
190 Y=Y-1
200 LINPUT #1:M$
210 FOR J=1 TO Y :: IF POS(M$,K$(J),1)=0 THEN 290
220 IF W$(J,1)=CHR$(127)THEN 250
230 W=W+1 :: IF W$(J,W)=CHR$(127)THEN W=0 :: GOTO 290
240 IF POS(M$,W$(J,W),1)=0 THEN 230
250 IF Q>1 THEN PRINT #2:M$
260 IF Q<>2 THEN PRINT M$
270 IF S=1 THEN 310
280 IF W$(J,W)<>CHR$(127)THEN 230
290 NEXT J
300 IF EOF(1)<>1 THEN 200
310 CLOSE #1 :: DISPLAY AT(24,1):"FINISHED - PRESS ANY
    KEY" :: CALL SOUND(200,500,5)
320 CALL KEY(0,K,ST):: IF ST=0 THEN 320 ELSE CALL CLEAR
    :: GOTO 110
```

You can set up a key file in TI-Writer, just remember that each 80-character line is a separate record, and keep the Alpha Lock down! However, this is the program that I plan to use to set up a key file index of all the newsletters you have sent me, if I ever find the time -

```
100 DISPLAY AT(3,10)ERASE ALL:"TIGERCUB": :"   KEYWORD
    INDEX WRITER" !by Jim Peterson
110 DISPLAY AT(8,1):"Filename? DSK" ::
    ACCEPT AT(8,14):F$ :: OPEN #1:"DSK"&F$,APPEND ::
    CALL KEY(3,K,S)
120 P$="*****" :: Y=00 :: M$="**" :: P=00
130 DISPLAY AT(12,1):"NEWSLETTER? ":P$ ::
    ACCEPT AT(13,1)SIZE(-28):P$ :: IF SEG$(P$,1,3)="END"
    THEN CLOSE #1 :: STOP
140 DISPLAY AT(14,1):"YEAR?";Y :: ACCEPT
    AT(14,7)VALIDATE(DIGIT)SIZE(-4):Y
150 DISPLAY AT(14,13):"MONTH? "&M$ ::
    ACCEPT AT(14,20)SIZE(-9):M$
160 DISPLAY AT(16,1):"PAGE?";P :: ACCEPT
    AT(16,7)VALIDATE(DIGIT)SIZE(-3):P
170 DISPLAY AT(18,1):"ARTICLE? " :: ACCEPT AT(19,1):A$
180 DISPLAY AT(20,1):"AUTHOR?" :: ACCEPT AT(21,1):AU$
190 DISPLAY AT(22,1):"KEYWORDS?" :: ACCEPT AT(23,1):K$
200 PRINT #1:P$&" "&STR$(Y)&" "&M$&" "&STR$(P)&" "&A$&"
    "&AU$&" "&K$
210 GOTO 130
```

Here is one to have fun with, from an ingenious German programmer. I just could not resist adding a tuba to his band.

```
100 !BY TORSTEN NIEMIETZ, MARBACHER WEG 3,D-2800 BREMEN
    1,WEST GERMANY
110 FOR J=1 TO 10 :: READ T(J)
120 NEXT J :: E=330 :: A=440 :: H=494 :: C=554 :: K=659
    :: F=740 :: G=831
130 DISPLAY AT(3,8)ERASE ALL:"S - O - L - O":
    :TAB(10);"MIT OOMPAH": :RPT$("=",28): : :"BY":"
    TORSTEN NIEMIETZ": :"mit Oompah by Tigercub"
140 DISPLAY AT(18,1):"MAKE UP YOUR SOLO WITH":"KEYS 1 TO
    9 ... COME ON !!!"
150 FOR S=1 TO 2 :: CALL SOUND(200,E,3,H,3)::
    CALL SOUND(200,E,3,H,3)
160 CALL SOUND(200,E,3,C,3):: CALL SOUND(200,E,3,H,3)::
    NEXT S
170 M=E :: N=H :: O=C :: D=8 :: GOSUB 210 :: M=A :: N=K
    :: O=F :: D=4 :: GOSUB 210 :: M=E :: N=H :: O=C ::
    GOSUB 210 :: M=H :: N=F :: O=G :: D=2
180 GOSUB 210 :: M=A :: N=K :: O=F :: GOSUB 210 :: M=E
    :: N=H :: O=C :: GOSUB 210 :: M=H :: N=F :: O=G ::
    GOSUB 210
190 FOR X=10 TO 3 STEP -1 ::
    CALL SOUND(200,E,3,H,3,T(X),0)
200 NEXT X :: CALL SOUND(800,E,3,H,3,K,0):: GOTO 150
210 FOR X=1 TO D :: FOR Y=1 TO 2 :: GOSUB 280
220 CALL SOUND(200,M,3,N,3,T(R-48-(R=48))*.9375,30,-4,0)
230 NEXT Y :: GOSUB 280
240 CALL SOUND(200,M,3,O,3,T(R-48-(R=48))*.9375,30,-4,0)
    :: GOSUB 280
250 CALL SOUND(200,M,3,N,3,T(R-48-(R=48))*.9375,30,-4,0)
260 NEXT X :: RETURN
270 DATA 587,659,784,880,988,1175,1319,1568,1760,44733
280 CALL KEY(0,R,S):: IF S<>0 AND R>48 AND R<58 THEN
    RETURN ELSE R=57 :: RETURN
```

```
1 !ONE-LINER universal calendar for day of week of any
    date since 1905 - by Dennis Hodgson in Sydney News
    Digest
2 !input day, month, year as for instance 30,4,1986
100 A=1 :: INPUT D,M,Y :: FOR T=A TO M-A ::
    H=H+29+VAL(SEG$("202121221121",T,A)):: NEXT T ::
    J=H+(Y/4<>INT(Y/4)AND M>2)+365.25*INT((Y-A))+D ::
    PRINT SEG$("SASUMOTUWETHFR",2*(J-7*INT(J/7))+A,2)::
    RUN
```

Yes, there are legitimate uses for GRAM copiers and track copiers and such, but there is no way to get these utilities into the hands of the few who will only use them honestly, without also getting them into the hands of the many who will use them as burglar tools. And so, a few more nails are driven into the coffin...

Memory full, Jim Peterson o

Plotic

by D.N. Harris

I have found that most Robots tend to "speak" Plotic, which is a language dealing with travel and state of travel, and changes of tool. I have also found that any software that enables one to print control codes will make a suitable vehicle for PLOTIC.

One such vehicle is of course BASIC, in either TI BASIC or Extended BASIC. In this case the Plotic can go in as a series of DATA with a READ command and a PRINT #1 (or #255 if you wish!) that has the output channel defined as PIO. At the other end of the system you can have a laser cutter, or a diamond engraver, or a simple little plotter, the kind that always spits its pens into the works just when you want to pack it up, so that the neighbours can hear all those nice 4 letter words you had not realized were part of your reflexive vocabulary, especially if you have been programming half the night and frantically mean to go to bed and catch some sleep!

I refrained from mentioning Plotic until I had got the thing to draw 5 pointed stars, then an irregular shape such as a Chemical Retort, then my signature. Finally, it must be admitted that PLOTIC is an advanced graphics language that can be used to draw anything. As most Plotters have colour, quite startling multi-coloured effects are possible. I can try to send a program, or hand some in for down loading from the BBS but these programs include the code; CTRL[M] which makes your printer go to the beginning and list one line on top of the other, also CTRL[R] and CTRL[Q] which do some things like set ELITE mode on some printers! If you have a word-processing program you may be able to send these codes to your PLOTIC device. If not, BASIC and a nice chunk of 50 or so DATA statements will do. The segments are LINE segments.

continued on page 22

Calendar Programs

and no assembly

by Adrian Robinson, ROM Users Group, USA

This article has been in the back of my mind for several years. Calendars are a fairly frequent subject of programmers' efforts and, when new calendar programs appear, I often feel an urge to comment on them. Occasionally, a program may be founded on an incorrect algorithm, which results in inaccuracy of in a limited range of application. More often though the algorithm is correct, but extremely inefficient programming techniques are employed, which result in a program much longer than it needs to be. A calendar program can really be quite simple and I will discuss some simple techniques, but first I would like to say a little about the foundations for, and history of, the calendar. The history of development of the calendar is an interesting subject in itself.

A calendar is basically just an enumeration of the days in a year or of the number of rotations of the Earth about its axis during one revolution of the Earth around the Sun. Oddly enough there are at least three different definitions of the length of a year. They are the Sidereal Year, the Anomalistic Year and the Tropical Year. We will not discuss the first two of these but the Tropical Year is the year of the Seasons and thus the basis of the calendar. The Tropical Year is the time between two successive occurrences of the Vernal Equinox, the time when the Sun's apparent motion crosses the Earth's equatorial plane in its northward movement and marks the beginning of Spring. The length of the Tropical Year is approximately 365.2422 Mean Solar Days. We will refer to this value later.

As civilizations developed in various parts of the world, some of the earliest recorded information includes astronomical observations and the development of a calendar in one form or another. An observer may notice that the position of sunrise or sunset on the horizon varies, moving north an south, during the year and he may relate that cycle to the seasons. Then, by counting the days in the cycle he can forecast the seasons. A hunter-gatherer society can now decide its migration cycle or an agricultural society can decide when to plant and when to harvest its crops. Thus, even a crude calendar can be a powerful tool.

The roots of our calendar go back more than 7000 years. The ancient Egyptians first determined the length of the year as 360 days and then, centuries later as 365 days. With this value, in the year 4236 BC, they devised a calendar of twelve thirty day months supplemented by five consecutive year end holidays. Not a bad idea!! in fact, I think that I would prefer this to our current calendar with its months of oddly varying lengths. Much later, they realized that the year was more nearly 365.25 days in length and in 238 BC they provided for a leap year every forth year. This calendar, however, was not widely accepted.

Meanwhile the Romans were struggling still with a lunar calendar of twelve alternating 29 and 30 day months. (The cycle of lunar phases is 29.5 days). But twelve lunar months total only 354 days so they found it necessary to throw in an extra month about every two or three years to keep the seasons more or less in order.

Finally, in 46 BC Julius Caesar set about reforming the calendar. He made the calendar independent of the phases of the moon and changed the lengths of the months to total 365 days. He also adopted the Egyptians' 365.25 day year. In addition, he restored the time of the Vernal Equinox to its ancient date of March 25. Naturally this became the Julian Calendar. Aside from some inconsequential "political" adjustments the only significant change in the next sixteen centuries was the introduction early in the forth century AD of the Christian seven day week. This added a further complication to the calendar since the year is not divisible by seven.

Now the Julian calendar of 365.25 days differs from the actual length of the Tropical Year of 365.2422 days by about one day in every 128 years. As a result by the sixteenth century, the calendar date of the Vernal Equinox had advanced from March 25 to March 11. If this were to continue long enough we would have Spring in December and Winter in August (USA).

In 1582, therefore Pope Gregory XIII proceeded to correct the errors of the Julian calendar. You may have wondered why Pope Gregory would be concerned with the calendar. Well, the date of Easter was prescribed in AD 325 to be the First Sunday on or following the day of the first full moon following the Vernal Equinox. Hence he wanted to maintain a constant date of the Vernal Equinox. He therefore ordained that century years would not be counted as leap years unless they were also divisible by 400. This has the effect of making the average length of the calendar year -:

```
(365+.25)-(1/100)+(1/400) = 365.2425
```

which is a much better approximation to the true length of a year. He also made March 21 the date of the Vernal Equinox, as it was in AD 325 instead of the original March 25. Thus in 1582, in order to correct the calendar October 5 suddenly became October 15!

I think that it is worth noting that two contemporaries of Pope Gregory were Copernicus and Galileo, both of whom were persecuted by the Church for their "heretical beliefs" that the earth revolved around the sun and was not the center of the universe.

The world's acceptance of the Gregorian calendar has been slow. It took Great Britain and her Empire two centuries to accept the Gregorian calendar (1752). It seems that the venerable old Church of England had been very influential and it was not anxious to accept a calendar that was ordained by the Pope of Rome. However it is generally accepted today, at least in the western world. Although the Gregorian calendar has a residual error of one day in about 3300 years, it is a good enough approximation for many more centuries of practical use.

From time to time proposals are made for reform of the internal structure of the calendar. Although it still has twelve months, there is no longer any direct connection to the phases of the moon and the varying lengths of the months are actually illogical. The seven day week, however, has become an integral part of everyday life and would be very difficult to change.

My personal preference would be a year of thirteen identical months of twenty eight days, each of four weeks. Every month would begin on Sunday and end on Saturday. The year would be completed with an extra holiday (New Year's Eve?) following the thirteenth month. Leap years would have an additional extra holiday at the end. The current year 1989, would have been a good time to start this new calendar since it already started on a Sunday. The Vernal Equinox by the way, would occur on the 24th day of the third month, whatever its name might be. I am sure that there would be strong resistance from the Calendar publishing lobby since this would be a perpetual calendar. The calendar would be identical every year. There would never be a need to buy a new calendar except to get a new set of pictures!

In addition to the above reform, I would add a new term to the formula for the mean length of the calendar year-:

```
(365+.25)-(1/100)+(1/400)-(1/3200) = 365.2422 days
                                     (rounded).
```

The new term, -1/3200, means that years divisible by 3200 would not be counted as leap years, whereas in the Gregorian calendar they would be. The agreement of this value with the true length of the Tropical Year would allow this Robinsonian Calendar to serve without error for about 1000 centuries.

But wait!! This article was supposed to be primarily about calendar programming techniques. I guess I just got carried away. Let us reduce the calendar to its fundamentals. As indicated above, the calendar year is basically just a string of numbers from 1 to 365 (or 366). The division into months simply converts it to a concatenation of strings from 1 to 31, 1 to 28, etc.with allowance made for leap years.

Introduction of week days results in a need to "register" the date string to the repeated weekday string. But this simply requires that we determine the "offset" of January 1 from the first Sunday of the year. The rest of the year takes care of itself. If we think about these statements for just a moment we should see how very simple the problem is. The names and lengths of the months, as well as the weekdays, are predetermined. All we need to define a calendar, for any year, are two items of data-:

1. What is the offset for January 1st?
2. Is it or is it not a leap year?

The included program, for convenience actually computes the offset for each month individually. This way, a slight modification will allow the program to compute and print a single month.

So, finally, let us take a look at the program. Lines 18 and 20 contain the names and lengths of the months. Line 22 forms a string of weekdays and line 24 does the same for the dates 1 to 31 in four character blocks. Thus a week fills up BASIC's twenty eight column screen. Note the printer OPEN statement in line 28.

The mainline program starts with the loop at line 30, which reads the twelve months' names and lengths and calls the subprogram MONTH. MONTH then computes the number of days elapsed from a hypothetical zero date to the first day of the current month, allowing for all leap years. Line 48 then converts that to a modulo 7 number (0 to 6) which is the number of weekdays preceding the first date of the month. Line 32 adds a day to February if Y is a leap year. Line 34 uses L to take the proper segment of M$, and B to add the proper number of blanks to the front of M$ for the offset. Line 36 then displays and prints the month.

Lines 30 through 50, just twelve lines, comprise the essence of the program and it computes, displays and prints an accurate calendar for any year later than 1582, with six months per page. With relatively minor modifications the program could be tailored to print in any desired format.

```
10 ! SAVE DSK6.CALENDER
11 ! CALENDER PROGRAM
12 ! by Adrian Robinson
13 ! ROM Newsletter, August 1989
14 !
16 ON WARNING NEXT :: CALL CLEAR
18 DATA JANUARY,31,FEBRUARY,28,MARCH,31,APRIL,30,
    MAY,31,JUNE,30
20 DATA JULY,31,AUGUST,31,SEPTEMBER,30,OCTOBER,31,
    NOVEMBER,30,DECEMBER,31
22 W$=" SU  MO  TU  WE  TH  FR  SA "&RPT$("-",28)
24 M$="" :: FOR I=1 TO 31 ::
    M$=M$&RPT$(" ",3-LEN(STR$(I)))&STR$(I)&" " :: NEXT I
26 INPUT " YEAR: ":Y :: IF Y<1583 THEN
    PRINT "YEAR MUST BE LATER THAN 1582": : : GOTO 26
28 OPEN #1:"PIO",DISPLAY ,VARIABLE 28 :: PRINT #1:Y: :
30 FOR M=1 TO 12 :: READ MO$,L :: CALL MONTH((Y),M,B)
32 IF M=2 THEN IF (Y/4=INT(Y/4))-(Y/100=INT(Y/100))+
    (Y/400=INT(Y/400))THEN L=L+1
34 D$=RPT$("    ",B)&SEG$(M$,1,4*L)
36 PRINT " ";MO$:W$:D$: : :: PRINT #1:" ";MO$:W$:D$: :
37 IF M=6 THEN PRINT #1:CHR$(12): : :
38 NEXT M :: CLOSE #1 :: END
40 SUB MONTH(Y,M,B)
42 F=365*Y+31*(M-1)+1
44 IF M>2 THEN F=F-INT(2.3+.4*M)ELSE Y=Y-1
46 F=F+INT(Y/4)-INT(Y/100)+INT(Y/400)
48 F=F-1 :: B=F-7*INT(F/7)
50 SUBEND
```

```
100 ! SAVE DSK6.CALENDER1
110 ! CALENDER PROGRAM
120 ! by Adrian Robinson
130 ! ROM Newsletter, August 1989
140 ! Printout mods by Ross Mudie
150 DIM P$(12,9)
160 ON WARNING NEXT :: CALL CLEAR
170 DATA JANUARY,31,FEBRUARY,28,MARCH,31,APRIL,30,
    MAY,31,JUNE,30
180 DATA JULY,31,AUGUST,31,SEPTEMBER,30,OCTOBER,31,
    NOVEMBER,30,DECEMBER,31
190 W$=" SU  MO  TU  WE  TH  FR  SA "&RPT$("-",28)
200 M$="" :: FOR I=1 TO 31 ::
    M$=M$&RPT$(" ",3-LEN(STR$(I)))&STR$(I)&" " :: NEXT I
210 INPUT " YEAR: ":Y :: IF Y<1583 THEN PRINT
    "YEAR MUST BE LATER THAN 1582": : :: GOTO 210
220 OPEN #1:"PIO",DISPLAY ,VARIABLE ::
    PRINT #1:TAB(28);CHR$(14);Y:
230 FOR M=1 TO 12 :: READ MO$,L :: CALL MONTH((Y),M,B)
240 IF M=2 THEN IF (Y/4=INT(Y/4))-(Y/100=INT(Y/100))+
    (Y/400=INT(Y/400)) THEN L=L+1
250 D$=RPT$("    ",B)&SEG$(M$,1,4*L)
260 PRINT " ";MO$:W$:D$: : :
270 P$(M,1)=MO$ :: P$(M,2)=SEG$(W$,1,28)::
    P$(M,3)=SEG$(W$,29,28):: P$(M,4)=SEG$(D$,1,28)::
    P$(M,5)=SEG$(D$,29,28)
280 P$(M,6)=SEG$(D$,57,28):: P$(M,7)=SEG$(D$,85,28)::
    P$(M,8)=SEG$(D$,113,28)
290 IF LEN(D$)>141 THEN P$(M,9)=SEG$(D$,141,28)ELSE
    P$(M,9)=""
300 NEXT M
310 FOR M=1 TO 12 STEP 2
320 FOR L=1 TO 9
330 IF L=1 THEN PRINT #1:CHR$(14);TAB(3);P$(M,L);
    TAB(22);P$(M+1,L)
340 IF L>1 THEN PRINT #1:TAB(2);P$(M,L);TAB(40);
    P$(M+1,L)
350 NEXT L
360 PRINT #1: :
370 NEXT M :: PRINT #1:CHR$(12):: CLOSE #1 :: END
380 SUB MONTH(Y,M,B)
390 F=365*Y+31*(M-1)+1
400 IF M>2 THEN F=F-INT(2.3+.4*M)ELSE Y=Y-1
410 F=F+INT(Y/4)-INT(Y/100)+INT(Y/400)
420 F=F-1 :: B=F-7*INT(F/7)
430 SUBEND
```

continued from page 20 J480,0CTRL[M] WILL DRAW A STRAIGHT LINE. J480,0CTRL[M] J0,-480CTRL[M] J-480,0CTRL[M] J0,480CTRL[M] R0,-480 draws a box, but first you have to do a CTRL[R] and afterwards you have to do a CTRL[Q].

There are a lot of other commands to learn, various character sizes are possible, 3 on some plotters, 64 on the one I have, and multiple characters can be woven together, to create smooth 3 dimensional banners of superb quality. Here goes an attempt to send a 5 pointed star.

```
CTRL[Q]
HM
I240,-457
IM148,0
ID-148,108
ID-148,-108
ID57,174
ID-148,108
ID183,0
ID57,174
```

Starting again:

```
CTRL[Q]
HM240,-457CTRL[M]
IM148,0CTRL[M]
ID-148,108CTRL[M]
ID-148,-108CTRL[M]
ID57,174CTRL[M]
ID-148,108CTRL[M]
ID183,0CTRL[M]
ID57,174CTRL[M]
ID57,-174CTRL[M]
ID183,0CTRL[M]
ID-148,-108CTRL[M]
ID57,-174CTRL[M]CTRL[Q]
```

continued on page 31

Programming c99

part 4

by Craig Sheehan

So far in this series on programming with 'c99' we have covered the use of printf, numeric variables and sub-programs. Now is good time to review this material before progressing to this part, which deals with character variables and basic file handling techniques.

Character variables are very similar to integers and should not present any problems if you have followed the material so far. Just as integers hold a single value, character variables can hold only a single character. (To hold more than one character, arrays of characters are required. These will be discussed next month.) An example of how character variables are defined is given in Figure 7. It is basically the same as Figure 4 (see part 2, TND, September 1989) except it prints the date instead of the time and that the "%c" command is used in printf to display the separation character.

```
/* Routine to display the date. */

extern printf();

main()
  { int  date, month, year;
    char sep;

    /* Define the separation character */
    sep = '/';

    /* Set the date */
    date  = 26;
    month =  1;
    year  = 88;

    /* Print the date */
    printf("Today's date: %2d%c%2d%c%2d\n", date, sep,
      month, sep, year);

    exit(0);
  }
```

Figure 7 - Print the date.

When this program is run, it should produce the output "Today's date: 26/ 1/88". The character variable 'sep' is set by enclosing the required character in single quotes. As mentioned above, the "%c" sequence is used to display the separating character, but the sequence "%2d" used to display each integer is new. It operates in the same manner as "%d", except that the integer is printed so that it uses at least two characters, with preceding spaces added to the integer to make it large enough if necessary. For this reason, the month in the above example takes two spaces rather than one. Setting the minimum number of characters to print a variable in is useful for the production of neat tables. A final point to note is that the PRINTF command is split between two lines. Because 'c99' is free format, it does not matter if this is done, provided the line is split where a space would have occurred. It is even possible to leave several lines between the two halves of the statement.

An example of a use for character variables is for displaying the contents of a file on the screen. A program to accomplish this is shown below.

```
/* Program to display the contents of a display /   */
/*  variable 80 file on the screen.                 */
/*                                                  */
/* Source: Brain W. Kernighan, et al, "The C        */
/*   Programming Language", 2nd edition, 1988,      */
/*                          pages 16 to 17.         */
/*                                                  */
/* Modified for 'c99' by Craig Sheehan.  2/ 6/89    */
/*   21 Suzanne road, Mona Vale NSW 2103, Australia */

#include DSK1.STDIO
extern printf();

main()
  { char c;
    FILE inputf;

    inputf = fopen("DSK1.ECHO;C","r80");

    while ((c = getc(inputf)) != EOF)
      printf("%c", c);

    inputf = fclose(inputf);

    exit(0);
  }
```

Figure 8 - Echo a DV80 file to the screen

Before compiling this program, the following files from the 'c99' disk must included on your program disk: CSUP, CFIO, PRINTF and STDIO. To load the program, use the filenames: DSK1.CSUP, DSK1.CFIO, DSK1.PRINTF and finally the object file for the above program. The first thing in the program is a comment that details what the program does, where it came from and its history. As I mentioned last month, it is important to do this for all programs you write.

The next line contains an "#include" command which instructs the compiler to compile the contents of the named file before compiling the remainder of the program. It is equivalent to loading the file "STDIO" and typing the rest of the program around it. You may wish to load the file "STDIO" into the editor and examine it. Were you to do this, you would discover that it contains a collection of "extern" statements, which provide access to all of the file handling sub-programs in the CFIO library as well as some lines that start with "#define". "#define" commands are similar to the replace string function on TI-Writer's editor. As an example, one of these lines reads: "#define FILE int". Every occurrence of "FILE" in the 'c' program following this will be replaced with "int". Following the "#include" is an "extern", which should be familiar to you by now.

The actual program has been broken down to five distinct blocks. The first of these defines two variables: the character 'c' and the integer 'inputf'. Notice that "FILE" has been used instead of "int". As I mentioned in the preceding paragraph, every occurrence of "FILE" is changed into "int". Whilst using "int" in the first place would not affect the running of the program, I have used the word "FILE" so that is clear to someone reading this program that 'inputf' refers to an external file that is going to be used.

The second block opens the file. Full details of the CFIO library can be found in the 'c99' documentation. "fopen" performs the task of opening the file whose name is given as the first argument, in the format specified by the second argument. "r80" means to open a read only, display variable 80 file. "fopen" returns a number that identifies this file for use by other sub-programs in the CFIO library. This file number is stored in 'inputf' for future use.

The third block, translated into plain english, says: if we are not at the end of the file, print the next character on the screen. The key to understanding the while statement is to evaluate its innermost brackets first. The innermost brackets contain the statement:

```
c = getc(inputf)
```

"getc" retrieves the next character from the file defined by 'inputf'. This character is placed in the variable 'c'. The overall value of (c = getc(inputf)) is the value of 'c'. So after the next character has been retrieved, the while statement is equivalent to:

```
while ( c != EOF )
```

continued on page 17

c99 Quick Reference

Author unknown, USA

Command/Function	Description	Incl File	Obj. File
c=getchar();	Read character from keyboard		CSUP
c=putchar(c);	Write character to screen		CSUP
c=gets(buff);	Read a line from keyboard		CSUP
puts(string);	Write string to screen		CSUP
exit(c);	Exit the program		CSUP
abort(c);	Exit the program		CSUP
locate(row,col);	Locate cursor on screen		CSUP
key=poll(c);	Check keyboard status		CSUP
tscrn(f,b);	Change screen colour		CSUP
unit=fopen(name,mode);	Open a file	stdio	CFIO
c=fclose(unit);	Close a file	stdio	CFIO
c=getc(unit);	Read character from file	stdio	CFIO
c=putc(c,unit);	Write character to file	stdio	CFIO
c=fgets(buff,col,unit);	Read string from file	stdio	CFIO
c=fputs(string,unit);	Write string to file	stdio	CFIO
c=fread(buff,len,unit);	Read record from file	stdio	CFIO
c=fwrite(buff,len,unit);	Write record to file	stdio	CFIO
fseek(unit,recno);	Set record number	stdio	CFIO
fdelete(filename);	Delete a file	stdio	CFIO
c=feof(unit);	Test for end-of file	stdio	CFIO
c=ferrc(unit);	Get error code	stdio	CFIO
rewind(unit);	Rewind a file	stdio	CFIO
grfl();	Set to graphics 1 mode	grflrf	GRF1
text();	Set to text mode	grflrf	GRF1
screen(c);	Set screen colour to c	grflrf	GRF1
color(cs,f,b,);	Change colors for character set cs to f and b	grflrf	GRF1
chrdef(ch,str);	Define character patterns	grflrf	GRF1
chrset();	Load standard character patterns	grflrf	GRF1
patcpy(a,b);	Copy character pattern	grflrf	GRF1
clear();	Clear the screen	grflrf	GRF1
hchar(r,c,ch,n);	Place character n times horizontally	grflrf	GRF1
vchar(r,c,ch,n);	Place character n times vertically	grflrf	GRF1
c=gchar(r,c);	Return value of character at r c	grflrf	GRF1
s=joyst(u,&x,&y);	Read joystick u	grflrf	GRF1
c=key(u,&s);	Read keyboard u	grflrf	GRF1
sprite(spn,ch,col,dr,dc);	Define sprite	grflrf	GRF1
spdel(spn);	Delete sprite	grflrf	GRF1
spdall();	Delete all sprites	grflrf	GRF1
spcolr(spn,col);	Set sprite colour	grflrf	GRF1
sppat(spn,ch);	Set sprite pattern	grflrf	GRF1
sploct(spn,dr,dc);	Set sprite location	grflrf	GRF1
spmag(f);	Set sprite magnification	grflrf	GRF1
spmotn(spn,rv,cv);	Set sprite velocity	grflrf	GRF1
spmct(n);	Enable sprite auto-motion	grflrf	GRF1
spposn(spn,&rp,&cp);	Return sprite position	grflrf	GRF1
dsq=spdist(spn1,spn2);	Return distance between sprites	grflrf	GRF1
dsq=spdrc(spn,dr,dc,);	Return distance between sprite and location	grflrf	GRF1
flg=spcnc(spn1,spn2,tol);	Sprite coincidence	grflrf	GRF1
flg=spcrc(spn,dr,dc);	Coincidence sprite and location	grflrf	GRF1
flg=spcall();	Coincidence of all sprites	grflrf	GRF1
float number[FLOATLEN];	Define float type	floati	FLOAT
c=fpgets(s,f);	Prompt for floating point number	floati	FLOAT
fpput(f,s);	Display floating point number	floati	FLOAT
c=itof(i,f);	Converts integer to floating point	floati	FLOAT
i=ftoi(f);	Converts floating point to integer	floati	FLOAT
c=stof(s,f);	Converts string to floating point	floati	FLOAT
c=ftos(f,s,mode,sig,dec);	Float array to string array	floati	FLOAT
c=fexp(f1,op,f2,res);	Execute floating point expression	floati	FLOAT
c=fexp(f1,"+",f2,res);	Add two numbers	floati	FLOAT
c=fexp(f1,"-",f2,res);	Subtract two numbers	floati	FLOAT
c=fexp(f1,"*",f2,res);	Multiply two numbers	floati	FLOAT
c=fexp(f1,"/",f2,res);	Divide two numbers	floati	FLOAT
true=fcom(f1,rel,f2);	Compare two floating point numbers	floati	FLOAT
c=fint(f1,f2);	Returns greatest integer value	floati	FLOAT
c=fcopy(f1,f2);	Copy one float array to another	floati	FLOAT
filptr=topen(n,a,s);	Open a file (name,access,fsize)	tcioi	TCIO
eof=tread(b,r,f,&s);	Read a file (buff,rec,fileptr,&size)	tcioi	TCIO
eof=twrite(b,r,f,s);	Write a file (buff,rec,fileptr,size)	tcioi	TCIO
eof=tclose(fileptr);	Close a file	tcioi	TCIO
randomize();	Initialize random seed	random;c	
rndnum();	Generate a 16-bit random number	random;c	

continued on page 28

TI-Base Tutorial #5

by Martin Smoley, NorthCoast 99'ers

I am reserving the copyright on this material, but I will allow the copying of this material by anyone under the following conditions. (1) It must be copied in its entirety with no changes. (2) If it is retyped, credit must be given to myself and the NorthCoast 99ers, as above. (3) The last major condition is that there may not be any profit directly involved in the copying or transfer of this material. In other words, Clubs can use it in their newsletters and you can give a copy to your friends as long as it is free.

Well here it is December already. Gee! time flies when you are having fun. This month I am going to change my mind again. I said I did not like System type setups, so this month I am doing a system for you. This is my Version 1.02 and my 1988 finale. This tutorial will contain practically all programming, with only a couple comments from me. The whole thing works, so if this is what you wanted, your time is here. This type of program runs too slow for me, but once it is finished you can run the whole thing with a few number entries. TIBSYS, which is listed below, runs all the other command files (more or less). So, to get the system going you would type DO DSK2.TIBSYS <ENTER>. You can find your way through the system by the order of the DO DSKn.XXX commands. you will notice that all the remark statements are at the end of the command files. This is because the processor does not read anything after RETURN. Putting your remarks after that point will speed things up. I have also kept the size of the command files down so you can edit any of them using MODIFY COMMAND. I try to use the same field names (NM, FN, LN, MI) for all my data bases. This allows me to use this type of programming on several data bases by merely changing USE NEWNAMES to USE (WHATEVER). Except for possible minor field length problems this system command file should be usable for many things.

```
 LOCAL ? N 2 0
 LOCAL SEL N 2 0
REPLACE ? WITH 0
     DO DSK2.PREP1
     DO DSK2.SYSSCR
USE NEWNAMES
TOP
  WHILE .NOT. (?)
     DO DSK2.INFSCR1
     DO DSK2.SLCASE
  ENDWHILE
     DO DSK2.FIN1
RETURN
*
* TIBSYS        Save as TIBSYS/C
* ******    TI-Base System 12/1/88
*
 **********************************
```

Note: Do not type in the last two lines of each command file . I am referring to the *, and the **********s. I put those in to keep things separated.

```
CLEAR
*   Pre-Program Preparation
*
* PREP1  Save as  PREP1/C
* *****
*
CLOSE ALL
SET HEADING OFF
SET RECNUM OFF
COLOR WHITE DARK-BLUE
SET TALK OFF
WAIT 5
RETURN
*
 *************************
```

```
CLEAR
 COLOR BLACK GRAY
 WRITE 3,9,"This is a TI-Base System."
 WRITE 5,9,"It is a club type system"
 WRITE 7,9,"to produce a club Roster,"
 WRITE 9,9,"a complete set of labels,"
 WRITE 11,9,"or search for individual"
 WRITE 13,9,"names and print more than"
 WRITE 15,9,"one label for a specific"
 WRITE 17,9,"name on the list."
 WRITE 19,12,"** USEs NEWNAMES **"
  WAIT 4
  COLOR WHITE DARK-BLUE
RETURN
*
* SYSSCR        Save as SYSSCR/C
* ******    System Screen 12/1/88
*
 **********************************

CLEAR
   REPLACE SEL WITH -1
 WRITE 2,8,"** Make A Selection **"
 WRITE 4,10,"> 0 < To Quit command file"
 WRITE 6,10,"> 1 < Print Roster"
 WRITE 8,10,"> 2 < Print All Labels"
 WRITE 10,10,"> 3 < Print Spec. Labels"
 WRITE 12,10,"> 4 < Edit NEWNAMES"
 WRITE 14,10,"> 5 < Append To NEWNAMES"
WHILE (SEL<0) .OR. (SEL>5)
   WRITE 22,4,"Enter 0-5"
   READ 22,15,SEL
   WRITE 22,3,"                  "
ENDWHILE
CLEAR
RETURN
*
* INFSCR1      Save as INFSCR1/C
* *******   Info Screen 1  12/1/88
*
 **********************************

************************************

     DOCASE
       CASE SEL = 0
        WRITE 18,13,"Have a nice day"
        REPLACE ? WITH 1
        BREAK
       CASE SEL = 1
         DO DSK2.PRSTR
        BREAK
       CASE SEL = 2
         DO DSK2.LBLS5
        BREAK
       CASE SEL = 3
         DO DSK2.FNDPRNT1
        BREAK
       CASE SEL = 4
         DO DSK2.EDFL1
        BREAK
       CASE SEL = 5
         DO DSK2.APFL1
        BREAK
     ENDCASE
RETURN
*
* SLCASE        Save as SLCASE/C
* ******  Case Selection 12/1/88
*
 **********************************

SET PAGE=000
SET LINE=80
CLEAR
LOCAL TEMP C 60
LOCAL BLNK C 1
WRITE 10,4,"Set Printer + press ENTER"
READ 10,30,TEMP
CLEAR
WRITE 10,12,"Printing Roster"
SORT OFF
TOP
```

```
   REPLACE TEMP WITH "<27>E          ";
     | "          ";
     | "      ** NorthCoast Roster **"
     PRINT TEMP
     PRINT BLNK
SET LINE=134
PRINT ALL
SET LINE=80
   REPLACE TEMP WITH " <27>@ "
   PRINT TEMP
CLEAR
RETURN
*                    Version 1.02
* PRSTR       Save as PRSTR/C
* *****   Print Roster  12/03/88
*
 *********************************
SET PAGE=000
CLEAR
LOCAL TEMP C 40
LOCAL BLNK C 1
WRITE 10,4,"Set Printer + press ENTER"
READ 10,30,TEMP
CLEAR
WRITE 10,12,"Printing Labels"
SORT ON ZP
TOP
WHILE .NOT. (EOF)
   REPLACE TEMP WITH "<27>E          ";
     | "        Exp. Date " | XP
     PRINT TEMP
     PRINT BLNK
   REPLACE TEMP WITH TRIM(FN) | " ";
           | MI | " " | LN
     PRINT TEMP
     PRINT SA
   REPLACE TEMP WITH TRIM(CT) | " ";
           | ST | ". " | ZP
     PRINT TEMP
     PRINT BLNK
   MOVE
   ENDWHILE
CLEAR
RETURN
*
* LBLS5   Save as LBLS5/C 12/01/88
* *****       Prints all Labels
*
 *********************************

LOCAL SEL2 N 3 0
LOCAL MORE N 3 0
REPLACE MORE WITH 1
WHILE (MORE > 0)
  TOP
  DO DSK2.INFSCR2
 WRITE 19,6,"ENTER 1-5"
 READ 19,17,SEL2
     CLEAR
   WHILE (.NOT.(EOF)) .AND.;
     (NM <> SEL2)
    MOVE
   ENDWHILE
  IF (NM = SEL2)
    DO DSK2.DISPNA1
  ENDIF
 WRITE 6,6,"FIND  MORE  NAMES"
 WRITE 8,6,"0 = No    1 = Yes"
 READ 8,25,MORE
 CLEAR
ENDWHILE
RETURN
*
* FNDPRNT1  Save as FNDPRNT1/C
* ********            11/29/88
*
 ******************************

************************************

   LOCAL TEMP1 C 40
   LOCAL TEMP2 C 40
   LOCAL TEMP3 C 40
   LOCAL BLNK C 1
   LOCAL ANS N 3 0
     CLEAR
   REPLACE TEMP1 WITH "<27>E          ";
     | "  Exp. Date " | XP
     WRITE 10,3,TEMP1
   REPLACE TEMP2 WITH TRIM(FN) | " ";
           | MI | " " | LN
     WRITE 12,3,TEMP2
     WRITE 14,3,SA
   REPLACE TEMP3 WITH TRIM(CT) | " ";
           | ST | ". " | ZP
     WRITE 16,3,TEMP3
  WRITE 22,1,"   Number of Labels"
 READ 22,22,ANS
WRITE 22,1,"                          "
IF ANS > 0                            "
  DO DSK2.PR-LBLS1
ENDIF
 CLEAR
RETURN
*
* DISPNA1    Save as  DISPNA1/C
* *******           11/29/88
*
 ******************************

SET PAGE=000
SET LINE=80
WHILE (ANS > 0)
  WHILE (ANS > 0)
     PRINT TEMP1
     PRINT BLNK
     PRINT TEMP2
     PRINT SA
     PRINT TEMP3
     PRINT BLNK
    REPLACE ANS WITH ANS - 1
    WRITE 22,4,"  Labels To Go =",ANS
    WAIT 1
WRITE 22,4,"                          "
  ENDWHILE
    WRITE 22,4,"More? How many? "
   READ 22,22,ANS
WRITE 22,4,"                          "
ENDWHILE
 CLEAR
RETURN
*           Version 1.02  11/29/88
* PR-LBLS1  Save as PR-LBLS1/C
* ********  Multiple Label Print
*
 ******************************
CLEAR
COLOR WHITE MAGENTA
WRITE 10,8,"DataBase should be open."
SORT OFF
TOP
EDIT
WRITE 10,8,"DataBase is not closed!"
COLOR WHITE DARK-BLUE
RETURN
*
* EDFL1           Save as EDFL1/C
* *****     EDIT A File  12/02/88
*
 ******************************

CLEAR
COLOR WHITE MAGENTA
WRITE 10,8,"DataBase should be open."
SORT OFF
APPEND
WRITE 10,8,"DataBase is not closed!"
COLOR WHITE DARK-BLUE
RETURN
*
* APFL1          Save as APFL1/C
* *****      APPEND To  12/02/88
*
 ******************************

SET TALK ON
CLOSE ALL
SET HEADING ON
SET RECNUM ON
COLOR WHITE MAGENTA
```

continued on page 11

Forth to you too! Session 4

Author unknown

You now have a system disk which autoboots the options you selected and before you do anything else, again: make a backup disk! Believe me, this is no idle chatter. I have messed up quite a few disks with some ill defined or used word. While you are in the learning stage, making a backup is perhaps as important as getting familiar with Forth words and how they work. Take it from one who has spent a lot of time starting over (and over). Unlike TI-Writer, Forth does not have an Oops!, only a %! #$*. I use two drives with my write protected system disk in drive 1 and do my programming on a disk in drive 2.

If you work with one disk drive, your best bet is to get a copy of Doug Smith's "3-PASS DISK COPIER". This clever 2 screen program was published in the June 84 issue of Miller's Smart Programmer. It is handy even if you have 2 drives, because it shortens the time required to copy a Forth disk. With 2 drives you can also use the word FORTH-COPY (provided you booted -COPY). The disk to be copied must be in drive 2 and the blank disk in 1. It takes approximately 1 second per screen or a minute and a half to copy a disk. Since it is done a screen at a time your drives get a good workout. But be sure to initialize the disk first with n FORMAT-DISK, where n is the number of the drive you put the blank disk in. However, remember that Forth starts counting with 0 (zero). What you would normally call drive 1 is 0, 2 is 1, etc. 1 FORMAT-DISK initializes the disk in drive 2.

In the last session I touched briefly on the SWCH and UNSWCH words of the -PRINT option. I know you will have no trouble remembering them. SWCH n LIST UNSWCH will soon be as familiar to you as n EDIT. Try SWCH 3 LIST UNSWCH. If you did not forget to turn your printer on you got a listing of screen #3. This is much easier to read then 3 LIST, which puts the listing on your display, because the lines are not broken. Then there are TRIAD and TRIADS which are similar to LIST but have SWCH/UNSWCH built in. 31 TRIAD first looks for the next lower number than the one you gave which is divisible by three and then prints 3 screens. SWCH and UNSWCH are built in since you could not use TRIAD as a display command. TRIADS works the same way, except you specify a range of screens (n1 n2 TRIADS). It will print as many triads as are needed to cover the range you specified.

Tricky, eh? But very neat: 3 screens per page. And one more: INDEX. It does not include SWCH/UNSWCH because it can be used on the display, too. But it is one of my favourites. n1 n2 INDEX lists the 0 (zero) lines of the screens from n1 to n2. (SWCH 0 89 INDEX UNSWCH will get you a printed index of your whole FORTH disk.) Like LIST it is really better in printed form, because of the display being limited to 40 columns which makes it harder to read and digest.

When you start programming, keep a printed INDEX of your disk on hand and make sure you use the Forth convention of identifying your screens on line zero. I prefer to not only put a program name on line 0 but to include the needed load option(s) as well. For example:

```
SCR 28
( TEXT MODE SCREEN-DUMP          -PRINT )


HEX
: SCREEN-DUMP  SWCH
               03C0 00 DO I DUP 28 MOD 0=
                        IF CR THEN VSBR EMIT
                    LOOP CR UNSWCH ;
DECIMAL
```

I wrote this little routine to save my trials and tribulations before they scrolled off the display when working in the interactive mode.

One of the beauties of Forth is the opportunity to try definitions from the keyboard in the so-called I/A mode. You can define a word (: ----- ;) and, before you use it in a program, see if it will do what you had in mind. The problem is, as you keep goofing and trying again, they disappear off the top of your display. Unless your memory is a lot better than mine, you will find SCREEN-DUMP a helpful addition to your Forth vocabulary. Since we either do not need -64Support or made it part of our autoboot, I saved this routine on screen #28, then added 28 LOAD on screen #3. SCREEN-DUMP is booted along with the autoboot and available any time I need it.

Recap:

1. Make a backup of your working autoboot disk
2. -PRINT is one of the most useful load options. It provides some new words and makes some others more useful than they already are (LIST and INDEX)
3. Save and load the SCREEN-DUMP routine. It will help you by putting your experiments on paper for reference.

Get busy with Chapters 4 through 6 of Starting Forth. (You can skip 3 because the TI Forth editor is much better than what Brodie describes.) o

The (TI) Forth Dimension

by Jeff Stanford, JSC Users Group, USA

When TI-Forth became available at the beginning of last year, I purchased a copy for myself immediately. I had a prior experience with Forth. Before TI-Forth's release, a friend of mine purchased a copy of a Forth source listing. After entering it, he gave me a copy of his absolute (error free source code) to test and evaluate. His fig-Forth was fun to play with, but it lacked an editor and its disk interface was not very efficient or reliable. After a period of time, I determined his fig-Forth could provide tremendous power for my computer if it were only expanded. I then put it aside and returned to learning Assembly Language.

When Texas Instruments introduced TI-Forth, I had something to get excited about. With TI-Forth's power, what was taking me days to do in assembly language now only took mere hours. But even in this abundance of new power for the TI99/4A, a flaw still existed. The documentation included with TI-Forth, in classic Texas Instruments' form, was designed not to be an instructive text, but rather to serve as a reference manual for the experienced Forth programmer (a la Editor Assembler). At least, the writers had the decency to mention this and to give some source materials for the intrepid novice.

What I plan to do with this series of tutorials is to share my experiences with TI-Forth. The series of articles will take complete novices and introduce them to the power of this exciting language. To get my tutorial off to a good start, I have decided to cover the books that will be used as source materials first. This is not to imply that these are the best available, but rather they are the books that I have purchased and found useful to learn more about Forth.

Starting Forth, by Leo Brodie (Prentice-Hall)

This is, in my own opinion, the foremost text on Forth for both the beginner and the expert alike. Mr. Brodie writes in a very readable style. The cartoon characters he uses for each of the Forth words make great learning aids. The book is jam packed with general information as well as useful hints and footnotes. Differences between TI-Forth and standard Forth discussed by Brodie in the book are illustrated in the TI-Forth manual (Appendix C).

continued on page 14

Working with Numbers

by Joe Nollan, Tacoma Informer, USA

There are a number of statements in TI BASIC that help us deal with numbers. The math functions are pretty straight forward and covered well in the book. There are problems which are not algebraic functions but are more of a house keeping nature. The first number problem that I encountered was after dividing $17 between 3 people my answer had eight decimal places. This is a common problem when dealing with numbers. The solution is fairly simple; consider the number 5.66666667. To reduce this to two decimal places so that it can more easily represent dollars and cents, first multiply it by 100 to yield 566.666667. The next step makes use of the INT function which will eliminate the fractional portion of the number giving us 566. This can now be divided by 100 to yield 5.66. Although this process was explained in a step by step manner it can often be done in a single statement like X=(INT(100*X))/100.

Another common problem is rounding off. If you have a number like 5.98 and would like it rounded to the nearest dollar the following method can be used. First add 0.5 to the value and then use the INT function. In the example of 5.98, adding 0.5 to it will yield 6.48 and the INT function will reduce it to 6 even. If the original value was 5.14, adding 0.5 and using the INT function would reduce it to 5 even. The BASIC statement would look like this: X=INT(X+0.5).

There are a couple of statements that are invaluable when numbers are involved. The STR$(N) statement will convert the numeric variable N into a string variable. This can be used when printing the value in a statement. When printed as a string, you will not have an extra space ahead of the number. The inverse of the STR$ function is the VAL(N$) statement. This statement will convert a string value into a numeric expression. You can not perform any calculations on a numeric expression whilst it is in string format.

A big word with numbers is truncation (cutting short). This shows its ugly head when you print out a column of numbers and the trailing zeros are left off, making your column look like it was done by a five year old. The problem can be solved by converting the numbers to string variables and printing them as such. The trailing zero problem can be solved by adding a small amount to the number. Working with dollars and cents, you may have a number like $9.50 that will look like $9.5 when printed. For example let us start with a numeric variable X and use it to represent the cost of an item (in dollars and cents). The $9.50 will be used as the value for this example. First we add a small amount to the value which will not affect the original number. In this case we are dealing with two decimal places so we will add a three decimal place number to it. X+0.001 will do it and we now have a value of 9.501. The last digit of 1 will hold the zero next to it. Before this figure is printed it is first converted to a string, then printed without the last character. First use X$=STR$(X) to convert it to a string variable. Now we will use the SEG$ function to drop the last character. The SEG$ function requires a string (X$), a point to start in the string (1) and how many characters (LEN(X$)-1). The BASIC statement would be: X$=SEG$(X$,1,LEN(X$)-1). With these values the SEG$ function will take all of the X$ starting from the first character and include everything but the last character. Our new string will contain 9.50. The value can then be printed in a sentence with a statement like PRINT "THE ITEM COST IS $";Z$;" WHOLESALE".

Printing a column of numbers presents another problem. Tab settings will line up the left digits which is great if the values all have the same number of digits and awful if they do not. One way to solve this problem is to use the LEN(X$) statement to determine the TAB(Z) value. Look at this example: PRINT "ITEM COST";TAB(15-LEN(X$));X$. In this example the larger the number, the smaller the TAB will be and the column will be lined up on the right.

These are not the only ways to solve number problems; Extended BASIC has ways around them as well. This is intended to give some ideas so that you can solve your own specific problems.

From The Tacoma Informer, August 1989, retyped and edited by Ross Mudie. o

continued from page 24

rnd(n);	Generate a random number between 0 and n-1	random;c
n=atoi(s);	Convert string to integer	conv;c
s=itod(nbr,str,sz);	Convert number to signed decimal	conv;c
n=xtoi(hexstr,nbr);	Convert hexstring to integer	conv;c
bitmap(fore,back);	Change to bitmapped screen mode	biti BITSUP
bitclr();	Clears the entire screen	biti BITSUP
plot(x,y,c,t,);	Turns on single pixel	biti BITSUP
line(x1,y1,x2,y2,c,t);	Draws line between two points	biti BITSUP
rect(x1,y1,x2,y2,c,t);	Draws a rectangle	biti BITSUP
circle(xc,yc,r,c,t);	Draws a circle	biti BITSUP
bitxt();	Copies ASCII characters into CPU RAM	biti BITSUP
bputch(ASCII,r,c,col);	Similar to putchar()	biti BITSUP
bputs(r,c,col,str);	Similar to puts()	biti BITSUP
blanks(r,c);	Places a blank on screen	biti BITSUP
btblanks(r,c,count);	Blanks sequence of locations	biti BITSUP
bgetch(r,c,col);	Returns keypress of user input	biti BITSUP
bgets(buffadr,s,r,c,col);	Inserts characters in buffer	biti BITSUP
getky();	Scans keyboard similar to poll()	biti BITSUP

Notes:

The purpose of "c99 Quick Reference" is to provide a handy summary of c99 command syntax and required parameters, a brief dscription and a reference to "include" and "object" files required to support a particular command. All references were re-capped from Clint Pulley's release diskette for c99 Version 2.0 except for "biti" and "bitsup" which are based on Jay Holovacs BITRTN and BITWRT Rel. 2.0.

By necessity the description of the commands had to be brief and is intended to be more of a "memory jogger". In all cases the user is urged to refer to the full documentation for all items.

The naming of include and object files reflect the preference of the compiler of this quick reference. You may have your own system and can feel free to use any suitable editor to make necessary changes. o

Rambles

by Stephen Shaw, England

News is in that Adelaide TI Computer Club are tackling the problem of getting RGB output from the TI99/4A in a very technical manner indeed. Previous published circuits have not resulted in pleasing results, as the RGB signal is obtained by "undoing" the processing which results in the standard colour difference signal output, resulting in an output which has been processed twice, rather than not at all! ATICC have decided to produce their own monolithic circuit to do the job, with computer checked circuits no less, with RGB TTL and Analog output for about A$85.

I suspect most members will not be aware of the work involved in producing a short-run VLSI chip, but our technically minded members should be suitably impressed. If you would like more information , write to:

Colin Cartwright, c/o Fred Cugley,
26 Suffolk Ave, BRAHMA LODGE, SA, AUSTRALIA, 5109.

==========

Report from Geoff Phillips of Australia that the Horizon RAMdisk, using OS 7.3, will not handle TI-Artist or TI-Pascal (referring possibly to USCD Pascal?), while one of our own members reports that TI-Base is unhappy with the HRD.

==========

MICROpendium December 1988 had a useful article on connecting disk drives to the TI99/4A system, with much detail on whether you should or should not use resistor termination packs, whether additional or different value resistors were appropriate, and the different ways of selecting drives. TEAC drives are specifically mentioned. Good article. If you do not subscribe take out a subscription now and ask for the back issue!

==========

A belated welcome back to the group to C H Street who asked what was required to attach a printer to his console, and also asked about modems.

i. A modem is used to connect a telephone line to the computer, and in addition to a modem you will require an RS232 device, usually a card for the peripheral expansion box.
ii. Most printers these days come with a parallel interface and with a serial interface available as an option. This refers to the plug on the printer side as well as to the format that the printer will accept the information in.

The bare console does not have any suitable output socket and you must purchase a separate device to attach to the computer, which is either a standalone device, or a card for the peripheral expansion box.

(A remarkable number of owners have purchased an expensive printer and then tried to connect it directly to their console! And having spent all their money on a printer, cannot afford a peripheral device to drive it.)

Standalones tend to be parallel only, while the cards often offer a parallel port as well as one or two serial ports. Connecting the output port to the printer will require a specially made cable. Have a word with your supplier and/or our own Mike Goddard before laying out your money! Make sure you see the printer working with your computer before laying down the cash! There are two types of parallel output, Epson and Centronix, although the term Centronix is often used when Epson is meant! The difference is a major one if you wish to use the TI RS232 peripheral card, as it does not drive a printer with a true Centronix port (such as the Tandy range) without some additional circuitry to provide the signals the Centronix port requires!

A few printers use an "inverted data strobe" (no I have no idea whether it is catching) such as the Hewlett Packard Ink Jet (got one going spare?) and this also needs a little circuitry to interface to the TI card correctly.

Wiring (for Epson, may be OK on others) of the parallel port is:

COMPUTER: PRINTER:
Pins 1 to 9 Pins 1 to 9 (connect 1 to 1 etc)
Pin 10, connect to Pin 11
TI pins 12 to 15 are not used.
TI pins 11 and 16 are "logic ground" and may usually be connected to printer pins 19 to 30. (Not to be connected to printer "chassis".)

The TI peripheral (and most TI99/4A software) was designed for the Epson printer range and Epson printers do not provide any problems at all. Other printers may not be fully compatible for graphics dumps. In addition to the peripherals, there are also modules available from Databiotics which have parallel printer ribbons coming out of the actual module! These are spread sheet and word processing modules.

Other printer considerations, apart from graphics compatibility and of course print quality and noise, are serviceability; how easily can you obtain a new print head or new ribbons and how much are the ribbons and how long do they last? I recall a TI99/4A owner who got a good printer bargain only to discover the ribbons were expensive and did not last too long and too many printer purchasers discover they cannot get replacement ribbons at all. Epson ribbons are very widely available!

==========

What word processors are available?

TI-Writer is the standard, now upgraded to Funnelweb, both on disk. Press is an upcoming program which may be a replacement. If you have no disk, then Databiotics have a module which can save and load to tape, and there is the option with that of a printer cable coming from the module.

==========

How to Edit a BASIC Program:

From Nicky Goddard, Age 9, thank you Nicky.

Nicky wrote to me regarding a disk AUTOLOAD program which allowed you to input a 1, 2 or a 3 for drive number. Nicky pointed out that the TI disk controller card could be modified to operate 4 drives as can the Myarc and CorComp cards, while RAMdisks can take higher disk numbers.

How to edit the LOAD program:
Type EDIT 1 and press FCTN[X] until you see on screen:
""DISK? 1-3" and change the 3 to whatever number you wish. Press FCTN[X] more times until you read on screen ("123") and add your extra number(s) using INSERT (FCTN[2]). Now press ENTER and save your program and there you are.

==========

Nicky also asks who wrote the disk AUTOLOAD program. This appears in many formats, which all have in common a line something like "DSKn.1234567890" or something equally odd.

This originated in issue 2 of 99er Magazine back in July 1981. Lots of people have decorated the program since and it can be found with lots of extras nowadays, but the essential part is unchanged and involves "a program that writes the program", as the program name is INSERTed into that odd line by the program itself. The author? Charles Ehninger, who went on to found Futura Software, a company which advertised a lot but did not obviously sell too much and is not too well known today (no longer trading of course!).

The part of the program that reads the disk directory comes from TI and can be found in the Disk Controller documentation.

continued from page 1

Then there was the console from a school which was diagnosed as having a faulty processor at the console repair night. Dick had removed the processor and when I tried it in my console it was OK. Looking at it later, with the processor soldered back in, I found that the READY line to the processor was low so that the prcessor was held in its wait state. This means that the LOAD interrupt does not operate, as it requires the IAQ signal, and the console tester will not run, unless it starts from power up, before any GROM accesses. The READY line is mainly controlled by the GROMs so I removed these and still the READY line stayed low. Another chip that controls the READY line is the sound chip so I removed that also and now the console tester worked. One of the first things the console tester does is to turn off the sound chip. So what I had found was that any of the GROM chips or sound chip in place and the READY line was held low. On closer examination of the board, I could see a track that was passing between two holes of the processor had been pushed sideways and was touching the pad round one of the holes. No doubt damaged in the unsoldering process, as others were also in a similar state but not actually touching. Once that was cleared, I could replace all but the system GROM, which must have been the original fault, as it would not release the READY line. A new GROM installed and good as new! Two other consoles looked at seemed to have bad processors, one of which was removed and found to be not working. Robert revived a bad key on a keyboard and I have subsequently fixed an Extended BASIC module with a bad ROM. By the way, the Extended BASIC module has two ROMs in it, a 4K byte one (addresses >6000 to >6FFF) and an 8K byte one which is bank switched to cover the addresses >7000 to >7FFF. The bank switching is done by writing to an address like >7000 to select one bank and to an address like >7002 to select the other bank.

****

Terry has received a letter from Stephen Shaw in England, one of our contributors (as he does to a large number of newsletters around the world) and obviously one of our readers, as he has picked up two errors in a music program published by us in October 1985. They occur in a program written by Fred Hawkins of the LeHigh group, based on some articles by Stephen, which generates music without any CALL SOUNDs and only two CALL LOADs to play the music. The lines to be corrected appear on page 13 of the October 1985 issue and should be:

590 DATA 0295D2010491D0AD1101
1120 DATA 04FF9FBFDFE0

Thank you Stephen, and if you should be talking to Peter Brooks at any time, give him my regards.

****

I have finally returned the hard disk on loan to me from John Vandermay for all these months (at least I gave it to John Paine to pass on). If you remember I had problems with sectors in the bit map area of the disk becoming unreadable and so I could not use Myarc's disk manager 5 on the disk and so could not copy files from it. If you want more details of the problems read past issues of TND. Garry Christensen of Brisbane wrote a quick back up program to allow the front half of a hard disk to be written to the back half of the same disk and his father Col, wrote a program called Hard Master to enable sectors of a hard disk to be read and written to. Unfortunately these did not do exactly what I wanted to do, as I wanted to copy files from one hard disk to another. so I modified Garry's program to write sectors from one hard disk to another, keeping the parameters of the read disk necessary to reconstruct the directory and file structure of the data, while retaining the parameters of the write disk so that it will operate correctly with the controller card. This will be the subject of a later article when I get a bit of time but I was wanting to report here that the transfer was successful and I transferred 20 megabytes of data from a 40 megabyte disk to a 20 megabyte disk in about 80 minutes. It then took me several hours to transfer files from one hard disk to another. I have recovered just about all the contents of the disk. There were 8 bad sectors, 2 of them were bit map sectors and the rest were File Descriptor sectors. So I have lost the information about 6 files, where they are and what their names are, but if I had the time I could probably recover most of that using Hard Master. By the way, there is a new version of Hard Master out which I must try and see if it will work with 10 megabyte disks.

****

One of the good things about TIsHUG, or indeed any of the user groups of the TI99/4A around the world I believe, is the willingness of members to help each other. I have experienced that myself when many members offered equipment to me at the beginning of this year to make the job of the editor easier and now I have had a response to my plea last month for someone to take over the onerous task of the editor. Bob Relyea spoke to me at the November meeting and has since come over to see what was involved. We hope to make a smooth transition during January unless there is someone else out there itching to have a go. Bob is a school teacher living at The Oaks which is out Picton way. We will be involved still by doing the paste-up and mailing, so that the work to be done is being spread around. Good on you Bob, welcome to the team! o

Newsletter Roundup

with Lou Amadio

Local Newsletters

Bug Bytes, October '89. A report on the Melbourne Faire, some Jim Peterson programs, and a programming challenge from Garry Christenson.

Melbourne Times, June/July '89. Peeping into Pascal by Peter Gleed, interesting article on the history of computers and structured BASIC with a sample program on circular maths. Melbourne Times, Aug/Sept '89. Structured maths and an Extended BASIC program on quadratics by Peter Gleed and a most unusual heat tester for the TI99/4A . More on Peeping into Pascal.

Overseas Newsletters

LA 99ers TopIcs, Sept '89. Release of TI-Base V2.02 which is compatible with hard disks, Hardmaster by Colin Christensen, a sector editor for hard disks, is now available through Asgard, next release of FirstBase on its way, Tenex winding down business for the TI99/4A, new "Harrison Word Processor" allows more pages of text and supports a menu driven interface versus the command language of TI-Writer, criticism of TI's early marketing policy with TI99/4A software, Beginning Forth #16 on sorting, "Hacker" program in Extended BASIC, novel way to change the screen in BASIC by Gene Bohot, how to unprotect an Extended BASIC or tape program and how to get 28 column listing to your printer. LA 99ers TopIcs, Oct '89. Mention of Graphic Editor and Hardmaster by Bill Gaskill, Chick De Marti's column and Beginning Forth column by Earl Raguse.

Spirit of 99, Oct '89. Number matrix programs in BASIC, TI-Sort for hard disks, AV-Indexer program for cassette and VCR labels, updated TI-Writer V4.2 with improved formatter, and informative BASIC programming by Irwin Hott.

ROM Sept '89. Justifying decimals in BASIC, Forth and TI-Writer tutorials, Multiplan investment model. ROM Oct '89. More Forth from Earl Raguse, using CTRL and FCTN keys in TI-Writer, tributes to Dr. Guy-Stefen Romano and John Guion and the constitution and articles of association of the group.

TIC TOC Rocky Mountain 99ers, Sept '89. Assembly programs to change Extended BASIC colours and provide true lower case, TI-Base tutorials and a listing of BASIC and Extended BASIC programs in club library.

TI Focus, Sept '89. New/updated software releases: Jiffy Card, Jiffy Flyer V3.02, Picture It, Giant Artist Posters, still no word on Press, new hard drives which auto park on shutdown, introduction to TI-Base, review of Infocom adventure games. TI Focus Oct '89. A review of XHi by Charles Good of the Lima group and other interesting general information. continued on page 3

Regional Group Reports

Meeting summary

Banana Coast	10/12/89	Sawtell
Carlingford	20/12/89	Carlingford
Central Coast	9/12/89	Saratoga
Glebe	14/12/89	Glebe
Illawarra	11/12/89	Keiraville
Liverpool	8/12/89	
Northern Suburbs	21/12/89	????
Sutherland	15/12/89	Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

Regular meetings are held in the Sawtell Tennis Club on the second Sunday of the month at 2 pm sharp. For information on meetings of the Banana Coast group, contact Kevin Cox at 7 Dewing Close, Bayldon, telephone (066)53 2649, or John Ryan of Mullaway via the BBS, user name SARA, or telephone (066)54 1451.

CARLINGFORD Regional Group

Regular meetings are normally on the third Wednesday of each month at 7.30pm. Contact Chris Buttner, 79 Jenkins Rd, Carlingford, (02)871 7753, for more information.

CENTRAL COAST Regional Group

Regular meetings are now normally held on the second Saturday of each month, 6.30pm at the home of John Goulton, 34 Mimosa Ave., Saratoga, (043)69 3990. Contact Russell Welham (043)92 4000.

Christmas Party at the December meeting

GLEBE Regional Group

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce St, Glebe. Contact Mike Slattery, (02)692 0559.

The September meeting of the Glebe Regional Group proved to be enlighting on the workings of the new TI video to TTL monitor conversion adaptors from South Australia. 3 types of monitors were available, as well as 3 technical people present to show them off to the best of their abilities.

As well as this, a program from up north was demonstrated, which showed that our little orphan is still being used in a commercial environment and will be for the next 5 years.

ILLAWARRA Regional Group

Regular meetings are normally on the third Monday of each month, except January, at 7.30pm, Keiraville Public School, Gipps Rd, Keiraville, opposite the Keiraville shopping centre. Contact Lou Amadio on (042)28 4906 for more information.

The December meeting this year will be on the second Monday of December, (11th), owing to the school holidays. This will be our Christmas party so bring your families and a plate of food for a fun time.

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIsHUG Sydney meeting at 7.30 pm. Contact Larry Saunders (02)644 7377 (home) or (02)642 7418 (work) for more information.

Press any day now, Legends II about the same (expanded to 3 disks now). All welcome.

Waiting on some new programs from the USA due any time.

No December meeting, next meeting 12th January 1990.

NORTHERN SUBURBS Regional Group

Regular meetings are held on the fourth Thursday of the month. If you want any information please ring Dennis Norman on (02)452 3920, or Dick Warburton on (02)918 8132.

Come and join in our fun. Dick Warburton.

SUTHERLAND Regional Group

Regular meetings are held on the third Friday of each month at the home of Peter Young, 51 Jannali Avenue, Jannali at 7.30pm. Group co-ordinator is Peter Young, (02) 528 8775. BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

The October meeting was well attended and proved to be a very interesting night. Much of the activity was of a technical nature. During the evening the group became occupied with:

- Running the console tester to diagnose problems with a faulty console for Kevin Taylor.
- Adjusting the rotational speed of my floppy disk drive using the Diagnostics Module.
- Rewiring Joe D'Ambra's modem cable and test driving his new modem in addition to Derek Wilkinson's recently acquired modem.
- Other assorted jobs including my battery charger and Garry Wilson's RAMdisk.

The latter part of the evening was spent in perusing the contents of the club BBS and down loading some software and information.

TIsHUG in Sydney

Monthly meetings start promptly at 2pm (except for full day tutorials) on the first Saturday of the month that is not part of a long weekend. They are held at the Woodstock Community Centre, Church street, Burwood. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software.

December 2 - Christmas Party at Woodstock.

Craig Sheehan (Meeting coordinator). o

continued from page 22

That is a better listing, but it should start with a CTRL[R] not a CTRL[Q].

For a bigger plotter all the numbers can be doubled to give a bigger 5 pointed star. The drawing of the star is a whiz-bang thing on the plotter. A dot matrix printer would go berserk and make a lot of ragged bits given the same task. It would also take a lot longer. It you did not know about Plotic then I suggest looking through the manuals for a lot of cutters, engravers, and plotting machines and you will discover that such a language exists. As to who invented it, perchance that could become a topic for study, but there it is, a way to do graphics with a resolution of 960 by 999 on a bigger plotter, or 480 by 999 on such a humble thing as I have been using. You can literally sign your name at that resolution! Graph paper and bigger scales (four or five times the intended drawing) and a sharp pencil help to plot graphics beyond the screen resolution of the computer.

If you want to plot something first trace it onto graph paper, then pick a number of points. For any straight line segment two points are enough, but to do a curve point density has to be increased where the curve approaches the vertical or horizontal to maintain a curve, so a circle will be circular not a round cornered square. Even plotting at 5 times the intended size with graph paper at the start, there is always some tricky segment to be smoothed out and a few extra points to be inserted.

Some plotters will do up to 80 characters a second, but try not to use them for word processing. A plotter is meant for pictures and diagrams, and labels where necessary. There are graphics typewriters which can be used to write letters but have an interface for use with a computer, and this form of a plotter can be the most satisfactory for those not already having some kind of plotter. o